A. PETRESCU

T. MOISA M. BROZICI C. PETRESCU V. LUNGU
N. ȚAPUȘ C. BERBECE TH. BALAN M. HĂNGANUȚ
D. GHEORGHIU D. POPESCU GH. PETRESCU A. PREDOI

MICROCALCULATOARELE FELIX M18, M18B, M118

Vol. II

AUTOMATICĂ
INFORMATICĂ

ELECTRONICĂ
MANAGEMENT

SERIA PRACTICĂ




AUTOMATICA
INFORMATICA
SERIA PRACTICĂ
ELECTRONICA
MANAGEMENT

SERIA PRACTICĂ
- Automatică
- Informatică
- Electronică
- Management

M. K. Starr : **Conducerea producţiei. Sisteme şi sinteze**
E. Crăciunoiu ş.a. : **Elemente de execuţie**
A. Vădescu ş.a. : **Radioreceptoare**
M. Mayer, G. Moltgen : **Tiristoarele în practică vol. I şi II.**
L. Zamfirescu şi I. Oprescu : **Automatizarea cuptoarelor industriale**
I. Papadache : **Automatica aplicată.** Ediţia I şi a II-a
Şt. Alexandru : **Automatizarea proceselor tehnologice în industria lemnului**
Lisicikin V. A. : **Prognoza tehnico-ştiinţifică în ramurile industriale**
G. Raymond : **Tehnica televiziunii în culori**
J. J. Samueli, ş.a. : **Instrumentaţia electronică în fizica nucleară**
Ţ. Homoş : **Capacitatea de producţie în construcţia de maşini**
S. Radu, D. Filoti : **Centrale telefonice automate**
M. Badea ş.a. : **Tranzistoare cu efect de cîmp**
D. N. Şapiro : **Proiectarea radioreceptoarelor**
V. Antonescu, M. Popovici : **Ghid pentru controlul statistic al calităţii**
V. Baltac ş.a. : **Calculatorul FELIX C-256. Structură şi programare**
G. Sonea, Sileţchi M. : **Creşterea planificată a productivităţii muncii**
R. L. Moriss : **Proiectarea cu circuite integrate TTL**
A. Brilliantov : **Calculul şi construcţia televizoarelor portabile**
Kaoru Iskikawa : **Controlul de calitate pentru maşini şi şefi de echipe**
Magnus Radke : **222 măsuri pentru reducerea costurilor**
I. Stăncioiu : **Eficienţa economică a asimilării de utilaje noi**
G. Lajtha : **Proiectarea reţelelor de telecomunicaţii**
Vătăşescu, A. ş.a. : **Dispozitive semiconductoare. Manual practic**
Ch. Jones : **Design : Metode şi aplicaţii**
E. S. Buffa : **Conducerea modernă a producţiei.** vol. I şi II
D. W. Davies ş.a. : **Reţele de interconectarea calculatoarelor**
Gh. Baştiurea : **Comanda numerică a maşinilor-unelte**
L. W. Crum : **Analiza valorii**
P. Foiaş : **Automatica şi informatica în procesele editorial-poligrafice**
P. Vezeanu, Şt. Pătraşcu : **Măsurarea temperaturii în tehnică**
T. Penescu, V. Petrescu : **Măsurarea presiunii în tehnică**
P. Popescu, P. Mihordea : **Măsurarea debitului în tehnică**
P. Vezeanu : **Măsurarea nivelului în tehnică**
A. Nadolo : **Măsurarea volumului şi cantităţii lichidelor în industrie**
C. Hidoş, P. Isac (coordonatori) : **Studiul muncii, I—VIII**
Hidoş, C. : **Analiza şi proiectarea circuitelor informaţionale**
Pisău, Gh., I. : **Elaborarea şi implementarea sistemelor informatice**
P. Constantinescu, V. Negoiţă : **Sistemele informatice modele ale conducerii**
V. Penescu, ş.a. : **Fişiere, baze şi bănci de date**
I. Ceauşu ş.a. : **SDV. Organizarea concepţiei, fabricaţiei, gestiunii**
S. Brebenel : **Practica transferului internaţional de tehnologie**
P. Constantinescu ş.a. : **Analiză, decizie, control**
A. Vătăşescu ş.a. : **Circuite integrate liniare, vol. 1 şi 2**
S. Maican : **Sisteme numerice cu circuite integrate**
I. Ristea ş.a. : **Manualul muncitorului electronist**
M. Florescu ş.a. : **Cibernetică, informatică, automatică în industria chimică**
E. Statnic, M. Gănescu : **Televizoare cu circuite integrate**
T. Geber ş.a. : **Echipamente periferice**
S. Călin ş.a. : **Optimizări în automatizări industriale**
M. Simionescu : **Proiectarea unitară a circuitelor electronice**
C. Cruceru : **Tehnica măsurărilor în telecomunicaţii**
P. Niţulescu : **Electroalimentarea instalaţiilor de telecomunicaţii**
R. Râpeanu ş.a. : **Circuite integrate analogice. Catalog**

Prof. dr. ing. **Adrian Petrescu**
– coordonator –
Dr. ing. **Trandafir Moisa**
Dr. ing. **Nicolae Țăpuș**
Ing. **Dan Gheorghiu**
Ing. **Mircea Brozici**
Dr. ing. **Constantin Berbece**
Dr. ing. **Damian Popescu**
Ing. **Constantin Petrescu**
Ing. **Theodor Bălan**
Ing. **Gheorghe Petrescu**
Ing. **Vasile Lungu**
Prof. dr. ing. **Marius Hăngănuț**
Ing. **Aristide Predoi**

Microcalculatoarele Felix M18, M18B, M118

Vol. 2



Editura Tehnică
București, 1984

Colectivul de elaborare al volumului 2 cuprinde specialiști de la Institutul Politehnic București, Întreprinderea de Calculatoare Electronice București, ICEMENERG, C.T.C.E. Suceava, Institutul Central pentru Conducere și Informatică, București, IPA București, IPA filiala Cluj-Napoca, respectiv Institutul Politehnic Cluj-Napoca.

Contribuția autorilor este următoarea :

A. PETRESCU : coordonare, 9.1, anexa 1 (continuare)
T. MOISA : cap. 7, 8, anexa 3
N. ȚĂPUȘ : cap. 6, anexa 2
D. GHEORGHIU : 9.2
M. BROZICI : 9.3
C. BERBECE : 9.4
D. POPESCU : 9.5, 9.6
C. PETRESCU : 9.7
Th. D. BĂLAN : 9.8
GH. PETRESCU : 9.9 (parțial)
V. LUNGU : 9.9 (parțial)
M. HĂNGĂNUȚ : 9.10 (parțial)
A. PREDOI : 9.10 (parțial)

Recenzie : dr. ing. **ADRIAN DAVIDOVICIU**

Redactor : ing. **PAUL ZAMFIRESCU**

Tehnoredactor : **STELA ȘERBĂNESCU**
Coperta : **SIMONA NICULESCU**

Bun de tipar 14.03.1984
Coli de tipar 16
CZ.62—50+681.14+658

Întreprinderea Poligrafică „Banat"
Timișoara, Calea Aradului nr. 1.
Republica Socialistă România.

Comanda nr. 191



Capitolul 6

Comenzile sistemului de operare SFDX-18

6.1. Introducere

Funcţiile generale ale sistemului de operare

Sistemul SFDX-18 este un sistem de operare care deserveşte microcalculatoarele FELIX M18/M18B/M118 în configuraţie cu disc flexibil şi minimum 32 Kocteţi memorie RAM.

Sistemul de operare este o colecţie de programe care asigură asistenţă utilizatorului în exploatarea şi întreţinerea resurselor sistemului de calcul.

Sistemul de operare SFDX-18 îndeplineşte următoarele funcţii generale :

— crearea şi modificarea fişierelor ;
— ştergerea fişierelor ;
— copierea fişierelor ;
— schimbarea numelui fişierelor ;
— actualizarea atributelor fişierelor ;
— conversia codului programelor obiect ;
— gestiunea resurselor ;
— combinarea modulelor de program în sisteme funcţionale complete ;
— execuţia programelor de sistem şi a programelor utilizator.

Utilizarea sistemului SFDX-18 presupune existenţa în memoria EPROM a unei versiuni de monitor, care să asigure operaţiile de I/E la nivel de caracter, în transfer programat (buclă de aşteptare) cu toate echipamentele periferice cu excepţia discului flexibil. SFDX-18 face apel la rutinele monitorului pentru realizarea operaţiilor de I/E.

Numele fişierelor

Sistemul de operare SFDX-18 lucrează cu fişiere ce pot să conţină :

— programe format sursă, obiect-relocatabil şi obiect-absolut ;
— date ;
— texte.

Felul datelor dintr-un fişier implică natura prelucrărilor asupra acestuia.

Numele unui fişier se specifică în mod unic prin :

[: < *dispozitiv* > :] *nume* [· < *extensie* >]

în cazul în care este rezident pe disc flexibil sau

: < *dispozitiv* > :

în cazul în care este rezident pe oricare alt suport (cartele, casetă magnetică, bandă magnetică etc.).

unde : < *nume* > este un identificator ce conține 1 : 6 caractere alfanumerice ce specifică numele fișierului respectiv ;

.<*extensie*> este un șir de maximum 3 caractere alfanumerice ce se asociază numelui de fișier. Extensia este opțională, însă dacă este folosită la crearea numelui fișierului, orice referire a acestuia trebuie să se facă utilizînd extensia. Este utilizată pentru a ușura identificarea diverselor forme ale aceluiași program. De exemplu programul TEST poate avea diverse forme :

TEST.ASM — programul sursă în limbaj de asamblare ;
TEST.REL — programul obiect-relocatabil ;
TEST.OBJ — programul obiect-absolut.

: < *dispozitiv* > : — specifică dispozitivul ce constituie suport pentru fișierul specificat (poate lipsi dacă fișierul se găsește pe discul flexibil 0).

Sistemul de operare SFDX-18 recunoaște următoarele nume de dispozitive periferice :

: F0 :, : F1 :, : F2 :, : F3 :	— unitățile de disc flexibil ;
: CI :	— intrare de la consola sistemului asignată curent ;
: CO :	— ieșire la consola sistemului asignată curent ;
: PR :	— cititor de bandă perforată ;
: PP :	— perforator de bandă ;
: CR :	— cititor de cartele ;
: LP :	— imprimantă ;
: VI :	— intrare de la USART ;
: VO :	— ieșire la USART ;
: TI :	— intrare de la interfața serială TTY ;
: TO :	— ieșire de la interfața serială TTY ;
: BB :	— (byte bucket) — dispozitiv inexistent dar care este tratat de SFDX drept unul fizic. Funcția lui este de a primi date, în scopul de a testa diverse mecanisme de transfer ;
: R1 :	— caseta magnetică ; deschiderea fișierului de intrare curent este realizată în prealabil sub MON18 ;
: P1 :	— caseta magnetică ; fișierul de ieșire curent este deschis în prealabil sub controlul MON18 (pe dispozitivul de perforare) ;
: L1 :	— caseta magnetică ; fișierul de ieșire curent este deschis în prealabil sub controlul MON18 (pe dispozitivul de listare) ;
: R2 :	— banda magnetică ; fișierul de intrare curent este deschis în prealabil sub controlul MON18 ;
: P2 :	— banda magnetică ; fișierul de ieșire curent este deschis în prealabil sub controlul MON18 (pe dispozitivul pe perforare) ;

:L2 : — banda magnetică ; fişierul de ieşire curent este deschis în prealabil sub controlul MON18 (pe dispozitivul de listare).

Dacă în anumite aplicaţii se doreşte utilizarea unor dispozitive nestandard în configuraţia sistemului, acestea se identifică prin următoarele nume :

:I1 : — citire de la consola nestandard.
:O1 : — scriere la consola nestandard ;

Consola nestandard trebuie asignată la CI respectiv la CO prin comanda CONSOL.

Numele fişierelor pot fi specificate şi generic. Aceasta se realizează prin utilizarea simbolurilor ? şi * pentru a înlocui unele caractere sau toate caracterele din numele fişierului.

Astfel : ? — înlocuieşte orice caracter alfanumeric ;
* — înlocuieşte orice grup de caractere alfanumerice.

Exemple :

:F1 : ASM80	— fişierul cu numele ASM80 de pe discul flexibil 1 ;
:F2 : PROG.FOR	— fişierul cu numele PROG.FOR de pe discul flexibil 2 ;
:CR :	— fişierul curent de la cititorul de cartele ;
:R2 :	— fişierul curent deschis pentru citire sub monitorul MON18 pe bandă magnetică ;
TEST.*	— specifică toate fişierele cu numele TEST indiferent de extensie ;
*.ASM	— specifică toate fişierele cu extensia ASM ;
AB ?.BAK	— specifică orice fişier cu extensia BAK şi numele format din trei caractere alfanumerice dintre care primele sînt AB ;
AB*.ASM	— specifică orice fişier al cărui nume are primele două caractere AB şi extensia ASM.

Atributele fişierelor

Fiecărui fişier îi sînt asociate 4 atribute ce pot fi controlate prin funcţia ATTRIB.

Cele patru atribute sînt :

I — invizibil ;

W — protejat la scriere (write protect) ;

F — format ;

S — sistem.

Fişierele cu **atributul I** (invizibil) nu sînt listate de comanda DIR decît dacă se specifică în mod explicit.

Fişierele cu **atributul W** (protejat la scriere) nu pot fi deschise pentru scriere sau punere la zi, nu pot fi şterse sau redenumite prin comenzi SFDX, sau apeluri de funcţii sistem SFDX din programele utilizator. Protecţia la scriere nu este activă în momentul în care se iniţializează discul cu comanda IDISK.

Fişierele cu **atributul F** (format) sînt copiate pe un nou disc flexibil, în mod implicit, în momentul iniţializării acestuia cu comanda IDISK.

Fişierele cu **atributul S** (sistem) sînt copiate pe un nou disc flexibil, în mod implicit, prin utilizarea comenzii COPY cu un anumit parametru. În acest fel se permite alegerea fişierelor ce se copiază pe un nou disc flexibil.

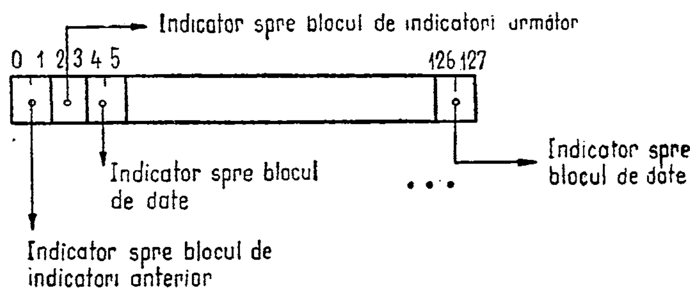


Fig. 6.1. Structura blocului de indicatori.

Un disc flexibil poate fi :

— **disc de manevră** — conține fișierele necesare întreținerii fișierului director și anume :

SFDX.TO

SFDX.LAB

SFDX.DIR

SFDX.MAP

— **disc sistem** — conține toate fișierele necesare sistemului de operare. Față de fișierele necesare discului de manevră mai cuprinde :

SFDX.BIN

SFDX.CLI

Un disc flexibil are sectorizare software conform formatului IBM 3740. Conține 77 de piste iar fiecare pistă conține 26 de sectoare (blocuri) a cîte 128 de octeți fiecare.

În total un disc flexibil conține 2002 blocuri (sectoare).

Alocarea spațiului pe disc se face în blocuri complete, deși ultimul bloc dintr-un fișier poate fi numai parțial utilizat. Un sector (bloc) nu este alocat niciodată la două fișiere distincte.

Structura generală a fișierelor

Fiecare sector (bloc) de pe disc are o adresă unică prin care poate fi referit. Adresa constă din doi octeți, unul reprezintă adresa pistei iar cel de al doilea reprezintă numărul sectorului. Pistele sînt numerotate de la 0 la 76 iar sectoarele de la 1 la 26.

Adresa unui bloc poartă și denumirea de indicator către acel bloc.

Există două tipuri de blocuri :

— bloc de indicatori (adrese), figura 6.1., conține adrese spre blocurile de date ;

Blocurile de indicatori sînt invizibile utilizatorului, ele conținînd informații necesare sistemului de operare.

— bloc de date, figura 6.2., conține informația ce se transferă în cadrul operațiilor de citire și scriere.



Fig. 6.2. Structura blocului de date.

Pentru fiecare 62 de blocuri de date este necesar un bloc de indicatori. Numărul total de blocuri (NB) necesar pentru un fișier de lungime L octeți se calculează astfel :

$$NB=[63\cdot[L/128/62]$$

Relația între blocurile de indicatori și cele de date este prezentată în figura 6.3.

Descrierea fișierelor de sistem

Așa cum am arătat anterior, orice disc flexibil conține patru fișiere de sistem, care sînt create în mod automat la inițializarea discului și anume :

SFDX.T0
SFDX.LAB
SFDX.DIR
SFDX.MAP

În continuare se vor prezenta structurile și funcțiile generale ale acestor fișiere.

SFDX.T0

Acest fișier conține un program denumit T0BOOT care se află pe pista 0. La inițializarea sistemului (acționarea comutatorului RESET/PROGLD sau LOAD de la panoul frontal) acest program se încarcă în memorie și se lansează în execuție.

Prin trecerea controlului programului T0BOOT, acesta va încărca restul sistemului de operare de pe disc. Are rolul de încărcător al sistemului de operare. Dacă discul nu este un disc sistem programul T0BOOT va da controlul monitorului.

SFDX.LAB

Acest fișier are lungimea de 128 de octeți și conține următoarea informație :

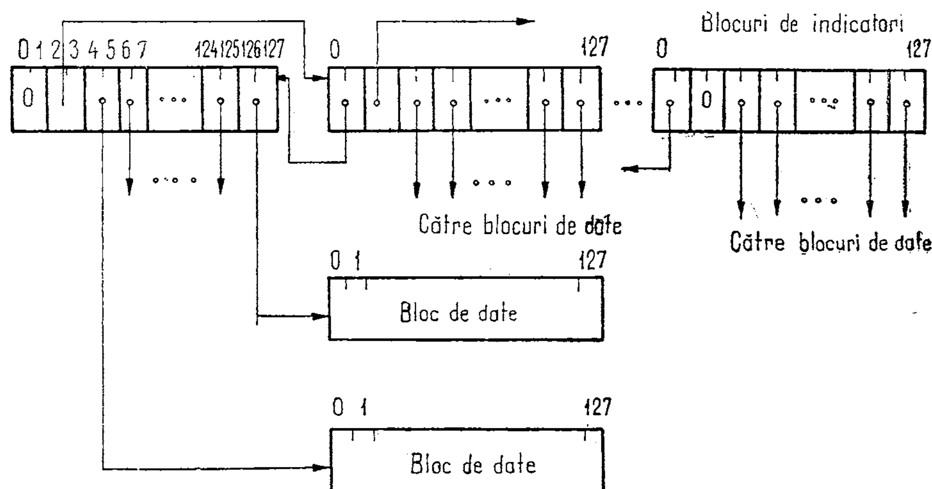


Fig. 6.3. Relația între blocurile de indicatori și cele de date.

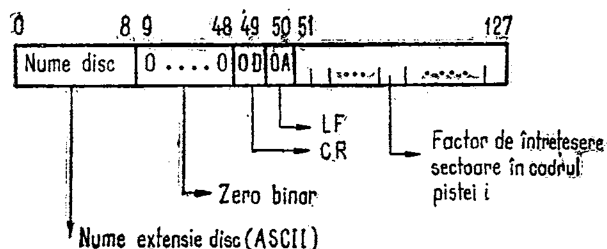


Fig. 6.4. Structura fişierului SFDX.LAB.

— primii 9 octeți ai acestui fişier conțin numele discului sub formă de caractere ASCII aaaaaabbb (aaaaaa-numele bbb extensia);
— următorii 40 de octeți sînt valori binare egale cu 0;
— fişierul conține în continuare caracterele CR=0DH şi LF=0AH după care conține 77 de octeți ce specifică factorul de întrețesere, a sectoarelor în cadrul pistei, pentru fiecare pistă.

În figura 6.4 se prezintă structura generală a acestui fişier.
Factorul de întrețesere este utilizat pentru a mări viteza de acces la blocurile de pe aceeaşi pistă.

În general între operațiile de citire a două blocuri se mai fac diverse prelucrări. Dacă blocurile sînt memorate în cadrul pistei unul după altul (adiacente) este posibil ca timpul necesar pentru prelucrările asupra unui bloc să fie suficient de mare pentru ca sectorul (blocul) următor să fi trecut de capul de citire. În acest caz este necesar să se aştepte o rotație completă a discului pentru a putea fi citit.

Acest dezavantaj se poate elimina dacă se specifică un factor de întrețesere a sectoarelor. Astfel dacă factorul de întrețesere este 2, blocurile cu adrese succesive sînt memorate fizic pe disc din două în două sectoare. În acest caz se pot citi toate blocurile în două rotații de disc față de 18 rotații cît ar fi implicat o memorare adiacentă.

Factorul de întrețesere este ales în funcție de timpul necesar operațiilor între citirea a două blocuri succesive.

SFDX.DIR

Acest fişier conține 25 de blocuri, fiecare din aceste blocuri putînd conține cîte 8 intrări în fişierul director.

Pentru fiecare fişier de pe disc se utilizează cîte o intrare în fişierul director. Astfel pe un disc flexibil pot exista maximum 200 de intrări, adică pot exista maximum 200 de fişiere.

O intrare în fişierul director are 16 octeți şi conține informații referitoare la:

— starea de alocare a intrării respective;
— numele fişierului;
— atributele fişierului;
— numărul de blocuri de date alocate fişierului;
— numărul de octeți în ultimul bloc;
— adresa către blocul de indicatori.

Structura unei intrări directoare este prezentată în figura 6.5.

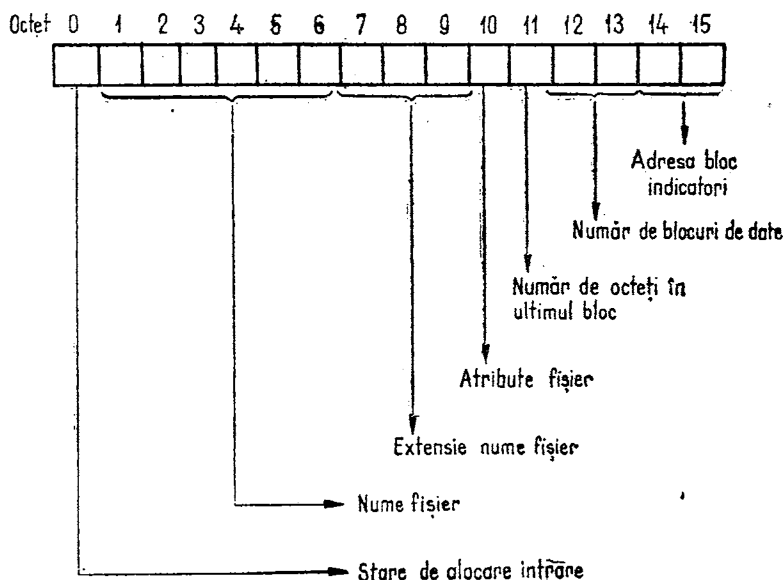


Fig. 6.5. Structura unei intrări în fişierul director.

Semnificaţia parametrilor din intrarea în fişierul director este următoarea :

— *stare alocare intrare* — este un parametru ce caracterizează intrarea şi poate avea una din valorile :

=0 — fişierul asociat cu această intrare este prezent pe disc.

=7FH — indică faptul că acestei intrări nu îi este asociat nici un fişier. Prima intrare în fişierul director cu valoarea 07FH indică sfîrşitul logic al fişierului director.

0FFH — indică faptul că fişierul asociat acestei intrări a existat odată pe disc, dar că în momentul de faţă este şters. Următorul fişier ce se va introduce în fişierul director va fi plasat pe prima intrare cu valoarea 0FFH.

— *nume fişier şi extensie* — reprezintă numele fişierului asociat cu această intrare. Numele şi extensia sînt păstrate ca o succesiune de coduri ASCII. Dacă numele sau extensia conţine mai puţin de 6 respectiv 3 caractere se face o aliniere la stînga şi se completează cu 0.

— *atribute fişier* — specifică atributele asociate fişierului respectiv. Semnificaţia biţilor acestui octet este următoarea :

bit0 = 0 — fişierul nu are atributul I (invizibil)
1 — fişierul are atributul I
bit1 = 0 — fişierul nu are atributul S (sistem)
1 — fişierul are atributul S
bit2 = 0 — fişierul nu are atributul W (protejat la scriere)
1 — fişierul are atributul W
bit7 = 0 — fişierul nu are atributul F (format)
1 — fişierul are atributul F

— *număr de octeți din ultimul bloc* — conține numărul ultimului octet de date din ultimul bloc al fișierului.

— *număr de blocuri de date* — conține numărul de blocuri de date din care este format fișierul.

— *adresă bloc indicatori* — este o adresă spre începutul blocurilor de indicatori asociați cu blocurile de date ale fișierului. Se specifică adresa de sector respectiv de pistă.

Sistemul găsește un fișier pe disc căutînd în fișierul director numele fișierului. Utilizînd adresa către blocul de indicatori, pe baza relației între blocurile de indicatori și cele de date (fig. 6.3), se ajunge la datele conținute în fișier.

SFDX.MAP

Acest fișier conține o hartă a alocării blocurilor pe disc. Fiecare bit din acest fișier reprezintă starea unui bloc (sector) de pe disc și anume:

0 — blocul respectiv este liber;

1 — blocul respectiv a fost alocat fie unui bloc de indicatori fie unui bloc de date.

Avînd în vedere faptul că orice disc, după inițializare are cele patru fișiere, acestea ocupînd 55 de sectoare (primele două piste și 3 sectoare de pe pista a treia), întotdeauna primii 55 de biți din fișierul SFDX.MAP au valoarea 1. Ultimii 46 biți sînt 0, deoarece există numai maxim 2002 blocuri pe un disc iar SFDX.MAP are 2048 de biți.

În figura 6.6 se prezintă relația între diferitele elemente ale fișierelor.

Editarea liniilor de comandă

Echipamentul periferic utilizat drept consolă constituie, în general, sursa comenzilor pentru sistemul de operare.

Comenzile pot fi preluate dintr-un fișier de pe disc dacă se folosește funcția SUBMIT a sistemului de operare.

Consola sistemului constituind sursa comenzilor, implică faptul ca fișierele asociate pseudonimelor dispozitivelor ce servesc ca intrare de la consolă :CI:, respectiv ieșire la consolă :CO:, să fie întotdeauna deschise.

Caracterele introduse la tastatura consolei sistemului sînt trimise, prin program, în ecou la dispozitivul de afișare al consolei. Caracterele introduse și caracterele trimise în ecou sînt memorate în Zone tampon separate. Lungimea maximă a zonei tampon este de 122 caractere, iar delimitarea comenzii se face de către < CR >.

Sistemul de operare analizează comanda numai după ce s-a introdus < CR >.

Linia de intrare, memorată în zona tampon asociată, poate fi editată înainte de a se introduce < CR >. Editarea se face utilizînd caractere de control (ce nu sînt introduse în Zona tampon de intrare) cum ar fi:

RUBOUT = 7FH — șterge din Zona tampon a liniei de intrare, ultimul caracter introdus. Pentru a indica anularea, se afișează la consolă caracterul șters din Zona tampon. Ștergerea ultimelor n caractere din linia de intrare se poate realiza prin acționarea de n ori a caracterului de control.

CTRL/X = 18H — anulează întregul conținut al liniei în curs de introducere. Se afișează caracterul # urmat de < CR > < LF >.

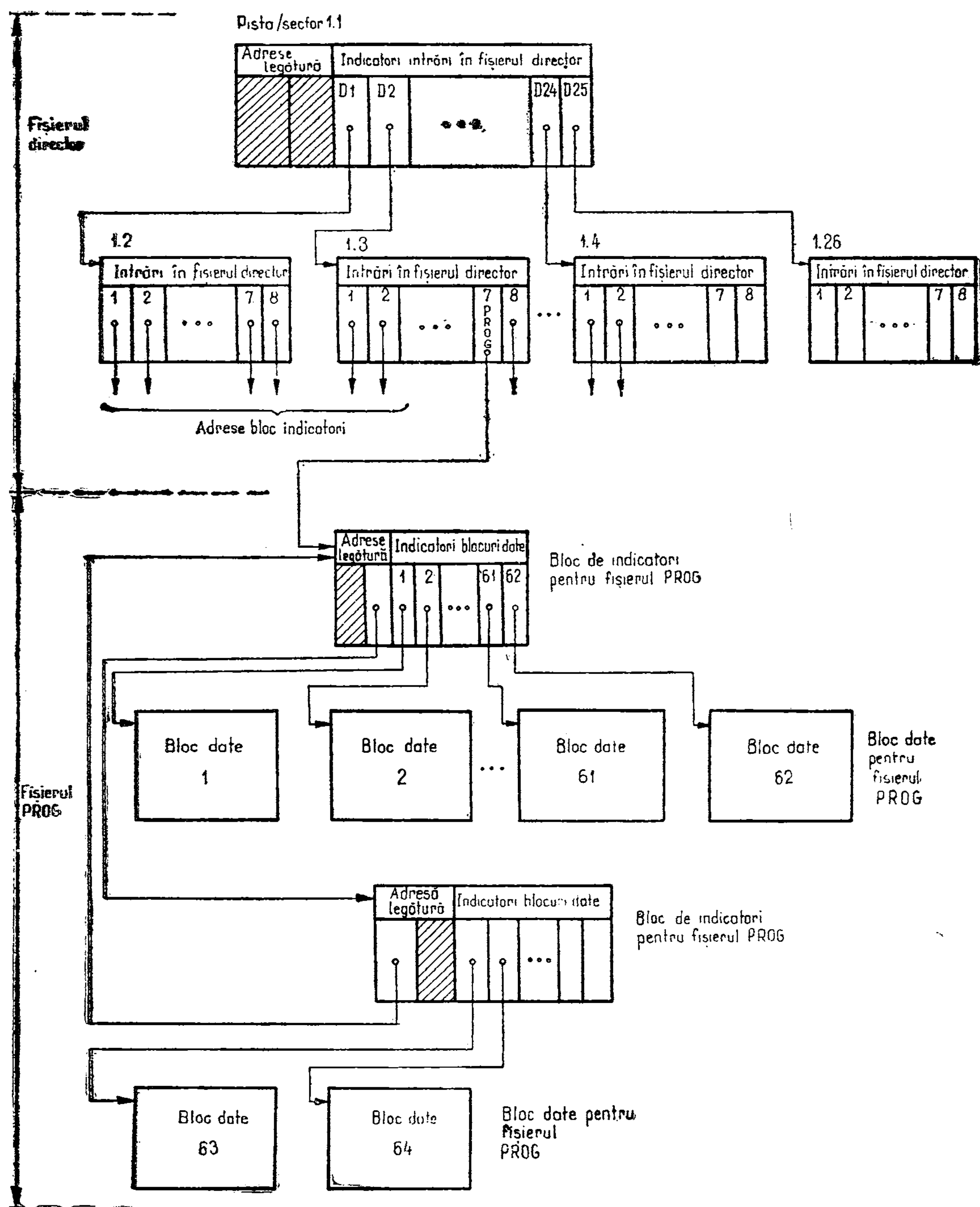


Fig. 6.6. Structura fişierelor pe disc.

CTRL/R = 12H — reafişează la consolă conţinutul curent al liniei în curs de introducere.
CTRL/S = 13H — opreşte listarea la consolă pînă cînd este introdus caracterul de control CTRL/Q.
CTRL/Q = 11H — reporneşte listarea curentă la consolă oprită prin CTRL/S.
CTRL/P = 10H — este utilizat înaintea caracterelor de control (inclusiv CTRL/P) pentru ca acestea să fie introduse în Zona tampon de intrare.
CTRL/Z = 1AH — introduce sfîrşit de fişier (end of file) în Zona tampon. Conţinutul Zonei tampon de intrare este şters. Cînd este introdus un sfîrşit de fişier, fişierele :CI: şi :CO: sînt închise, SFDX-18 este reîncărcat şi consola sistem iniţială este deschisă ca :CI: şi :CO:.

6.2. Descrierea comenzilor sistemului de operare SFDX-18

Comenzile sistemului de operare SFDX-18 sînt grupate în patru categorii de funcţii şi anume :

— pregătirea unui disc flexibil pentru utilizarea lui de către sistem ;

— crearea, ştergerea şi punerea la zi a fişierelor ;

— conversia caracterelor de codificare a programelor dintr-un format în altul ;

— executarea programelor.

6.2.1. Comanda de iniţializare a unui disc flexibil

Un disc flexibil nou trebuie să fie iniţializat înainte ca el să poată fi utilizat de SFDX-18. Iniţializarea sa poate fi făcută pentru a fi utilizat ca disc sistem sau disc de manevră. Comanda utilizată pentru iniţializare este :

IDISK — iniţializează un disc flexibil ca disc sistem sau de manevră.

Comanda IDISK

Formatul comenzii este următorul :

IDISK : < *dispozitiv* > : < *nume disc* > [· < *extensie* >] [S]

unde :

: < *dispozitiv* > : specifică unitatea de disc unde este introdus discul flexibil ce urmează să fie iniţializat.

< *nume disc* > reprezintă numele atribuit discului flexibil respectiv şi poate avea maxim 6 caractere.

· < *extensie* > reprezintă extensia asociată numelui şi poate avea maxim 3 caractere.

S este un parametru care, dacă este specificat, indică faptul că discul respectiv va fi inițializat ca disc sistem. Altfel acesta va fi inițializat ca disc de manevră.

Această comandă realizează inițializarea unui disc flexibil ca disc sistem, dacă parametrul S este specificat, sau ca disc de manevră.

Dacă se specifică unitatea de disc 0 (: F0 :) sistemul așteaptă schimbarea discului sistem cu cel ce se dorește să fie inițializat.

După inițializare, copierea fișierelor și a comenzilor necesare se face cu comanda COPY.

Exemple :

— *IDISK :F1 : NT01.0 S*
SYSTEM DISKETTE
—

Se inițializează discul din unitatea 1 ca disc sistem.

— *IDISK : F0 : NT01. 1*
LOAD OUTPUT DISKETTE, THEN TYPE (CR) < *CR* >
NON — SYSTEM DISKETTE
LOAD SYSTEM DISKETTE, THEN TYPE (CR) < *CR* >
—

Se inițializează discul din unitatea 0 ca disc de manevră. Procesul de inițializare implică schimbarea discului sistem, ce conține comanda IDISK, cu discul ce se inițializează conform cu indicațiile afișate la consolă.

6.2.2. Comenzile de control fișiere

Comenzile de control fișiere au rolul de a asigura funcțiile de :

— inventariere a informațiilor de pe un disc flexibil — **DIR**
— copiere a unor fișiere de pe un dispozitiv pe altul — **COPY**
— ștergerea unor fișiere — **DELETE**
— schimbare a atributelor unor fișiere — **ATTRIB**
— redenumire a numelui unui fișier — **RENAME**

Comanda DIR

Formatul comenzii este următorul :

DIR [**FOR** < *nume fișier* >] [**TO** <*nume fișier listing* >] [<*parametri* >]
unde :

< *nume fișier* > reprezintă fișierul (sau grupul de fișiere dacă numele acestora se specifică generic) ale cărui caracteristici se afișează ;

< *nume fișier listing* > reprezintă numele fișierului unde se depune informația referitoare la caracteristicile fișierului (fișierelor) specificat de nume fișier ;

< *parametri* > reprezintă parametri de control ai afișării. Pot lua valorile :

{1, 2, 3} — specifică unitatea de disc unde se caută < *nume fișier* >. Lipsa acestui parametru implică lucrul cu unitatea 0.

I — listează și caracteristicile fișierelor cu atributul invizibil. Lipsa acestui parametru face ca să nu aibă loc listarea caracteristicilor fișierelor invizibile.

F — listează numai numele fişierelor, fără caracteristicile lor.
P — operaţia de listare a caracteristicilor fişierului (fişierelor) se face aşteptîndu-se schimbarea discului.

Comanda **DIR** are rolul de a lista caracteristicile unor fişiere aflate pe unitatea de disc specificată, la echipamentul specificat.

Dacă **FOR** *<nume fişier>* este omis din cadrul comenzii, se face listarea tuturor fişierelor de pe unitatea de disc specificată (inclusiv cele cu atributul invizibil dacă parametrul I este specificat).

Dacă nu se specifică **TO** *< nume fişier listing >* afişarea caracteristicilor se face în mod implicit la consola sistemului.

Caracteristicile afişate în urma comenzii DIR sînt :
— nume fişier ;
— numărul de blocuri ocupate de fişierul respectiv ;
— atributele fişierului respectiv ;
— gradul de utilizare a discului.

Exemple

— **DIR FOR LINK. *** *< CR >*

DIRECTORY OF : F0 : PLM80.SYS

NAME .EXT BLKS LENGTH ATTR	NAME .EXT BLKS LENGTH ATTR
LINK 93 11521	LINK .OVL 26 3123
	119

1389/2002 BLOCKS USED

La consola sistemului s-au afişat caracteristicile fişierelor LINK şi LINK.OVL de pe unitatea de disc 0.

— **DIR TO : LP : FI** *< CR >*

DIRECTORY OF : F0 : PLM80.SYS

SFDX	.DIR	SFDX	.MAP
SFDX	.TO	SFDX	.LAB
SFDX	.BIN	SFDX	.CLI
COPY		PLM80	.OV0
PLM80	.OV1	PLM80	.OV2
PLM80	.OV3	PLM80	.OV4
PLM80	.LIB	LINK	
LINK	.OVL	LOCATE	
EDIT		PLM80	

1389/2002 BLOCKS USED

La imprimantă s-au afişat numele tuturor fişierelor de pe unitatea de disc 0.

— **DIR P I** *< CR >*

LOAD SOURCE DISKETTE TYPE < CR >
DIRECTORY OF : F0 : PLM80.SYS

NAME	.EXT	BLKS	LENGTH	ATTR	NAME	.EXT	BLKS	LENGTH	ATTR
SFDX	.DIR	26	3200	IF	SFDX	.MAP	3	256	IF
SFDX	.TO	24	2944	IF	SFDX	.LAB	2	128	IF
SFDX	.BIN	94	11740	SIF	SFDX	.CLI	21	2548	SIF
COPY		29	3550		PLM80	.OV0	149	18640	
PLM80	.OV1	230	28814		PLM80	.OV2	66	8121	
PLM80	.OV3	187	23478		PLM80	.OV4	65	7948	
PLM80	.LIB	45	5615		LINK		93	11521	
LINK	.OVL	26	3123		LOCATE		123	15370	
EDIT		59	7424		PLM80		170	21298	
					1389				

1389/2002 BLOCKS USED
LOAD SYSTEM DISKETTE, TYPE < CR >

La consola sistemului s-au afişat caracteristicile tuturor fişierelor, de pe unitatea 0.

Procesul de afişare, datorită parametrului P, cere schimbarea discului sistem ce conţine comanda DIR cu discul de pe care se doreşte afişarea caracteristicilor fişierelor. Operaţia de schimbare se face conform indicaţiilor afişate la consolă.

Comanda COPY

Formatul comenzii este următorul :

COPY < *nume fişier sursă* > [, < *nume fişier sursă* >]*
TO < *nume fişier destinaţie* > [< *parametri* >]

unde :

< *nume fişier sursă* > reprezintă numele fişierului sursă care se copiază ;

< *nume fişier destinaţie* > reprezintă numele fişierului destinaţie unde se copiază fişierul sursă (sau se copiază, concatenate, fişierele sursă) ;

< *parametri* > caracterizează modul în care va avea loc copierea fişierelor. Pot avea valorile :

C — parametru care face ca fişierul destinaţie să aibă aceleaşi atribute ca şi cel sursă. Dacă parametrul nu este specificat fişierul destinaţie nu va avea nici un atribut.

Q — parametru care cere din partea utilizatorului validarea operaţiei de copiere. La consola sistemului apare mesajul :
COPY < *nume fişier sursă* > **TO** < *nume fişier destinaţie* > ?
Dacă răspunsul este Y („yes") copierea are loc. La oricare alt răspuns nu execută copierea.

P — parametru care cere schimbarea discurilor flexibile în timpul procesului de copiere. Schimbarea se face conform dialogului :

LOAD SOURCE DISKETTE, THEN TYPE < CR >
LOAD OUTPUT DISKETTE, THEN TYPE < CR >
LOAD SYSTEM DISKETTE, THEN TYPE < CR >

S — parametru care realizează copierea numai a fişierelor cu atributul S în cazul specificării fişierelor sub formă generică.

N — parametru care realizează copierea numai a fişierelor nesistem (fără atributele S şi F) în cazul specificării fişierelor sub formă generică.

B — parametru care produce ştergerea automată a fişierului destinaţiei dacă acesta era deja existent pe disc şi copierea fişierului sursă specificat.

U — parametru ce specifică operaţia de punere la zi a fişierului destinaţie. Fişierul destinaţie se va completa cu conţinutul fişierului sursă. Dacă parametrul C nu este specificat, fişierul destinaţie va avea după operaţia de copiere aceleaşi atribute ca şi înainte. În cazul în care parametrul C este specificat fişierul destinaţie va primi în plus atributele fişierului sursă.

Comanda **COPY** copiază fişierul sau fişierele sursă în fişierul destinaţie specificat. Fişierele pot fi rezidente pe orice echipament periferic cu condiţia ca fişierul sursă care se copiază să fie fişier de intrare iar fişierul destinaţie să fie fişier de ieşire.

Dacă sînt specificate mai multe fişiere sursă, ele vor fi concatenate în fişierul destinaţie în ordinea specificării lor. Cînd se utilizează concatenarea numele fişierului destinaţie nu poate fi egal cu nici unul din nu-

mele fișierelor sursă. În cazul concatenării de fișiere, numele acestora trebuie să fie specificat integral (nu se admit nume generice).

În cazul în care numele fișierului sursă este identic cu numele fișierului destinație acesta din urmă poate fi specificat numai prin numele echipamentului periferic. Dacă numele fișierului sursă este la fel cu cel destinație și unitatea de disc este aceeași pentru ambele fișiere, aceasta implică schimbarea discului printr-un dialog cu utilizatorul ca cel prezentat la parametrul P.

În cazul în care fișierul destinație era deja existent, sistemul anunță utilizatorul prin mesajul :

< nume fișier destinație > FILE ALREADY EXISTS, DELETE ?

și acesta poate lua decizia de ștergere și copiere a fișierului sursă specificat introducînd Y <CR > sau de abandonare a procesului de copiere introducînd orice alt caracter (dacă nu a fost specificat cumva parametrii B sau U).

Exemple

Copierea fișierului BASIC de pe discul din unitatea 0 pe discul din unitatea 1, cu păstrarea atributelor, se poate face utilizînd una din comenzile :

— **COPY BASIC TO :F1:BASIC C** < *CR* >

sau :

— **COPY BASIC TO :F1: C** < *CR* >

Listarea conținutului unui fișier sursă la imprimantă se poate face utilizînd comanda :

— **COPY PROG.ASM TO :LP:**

Copierea a trei fișiere într-unul singur se face prin comanda :

— **COPY TEXT1, TEXT2, TEXT3, TO TEXT** < *CR* >

Copierea tuturor fișierelor sistem de pe discul 0 pe discul 1 se face utilizînd comanda :

— **COPY *.* TO :F1:** S < *CR* >

Copierea tuturor fișierelor de pe discul 0 pe discul 0, cu schimbarea discului (presupunînd că sistemul are numai o unitate de disc) se realizează prin comanda :

— **COPY *.* TO :F0: C** < *CR* >

LOAD SOURCE DISKETTE, THEN TYPE < CR >
LOAD OUTPUT DISKETTE, THEN TYPE < CR >
.
.
.
LOAD SYSTEM DISKETTE, THEN TYPE < CR >

Dacă fișierele care se copiază depășesc dimensiunea memoriei RAM disponibile, mesajele LOAD SOURCE și LOAD OUTPUT se afișează de mai multe ori, utilizatorul trebuind de fiecare dată să schimbe discul. După ce este copiat ultimul fișier este afișat mesajul LOAD SYSTEM.

După ce are loc copierea la consola sistemului se afișează :

COPIED < nume fișier sursă > TO < nume fișier destinație >

Comanda DELETE

Formatul comenzii este următorul :

DELETE < *nume fișier* > **[Q]** [, < *nume fișier* > **[Q]**]* **[P]**

unde :

< *nume fișier* > reprezintă numele fișierului (fișierelor) ce se șterg ca urmare a execuției comenzii.

Q — parametru care cere din partea utilizatorului validarea operației de ștergere fișier ;

P — parametru care cere schimbarea discurilor flexibile în timpul procesului de ștergere. Schimbarea se face conform dialogului

LOAD SOURCE DISKETTE, THEN TYPE < CR >
LOAD SYSTEM DISKETTE, THEN TYPE < CR >

Comanda **DELETE** șterge un fișier sau un grup de fișiere de pe disc, spațiul ocupat de aceste fișiere rămînînd disponibil pentru alte alocări.

După ce are loc operația de ștergere, la consola sistemului se afișează :

< *nume fișier* >, DELETED

Fișierele cu atributul W (protejat la scriere) nu pot fi șterse decît dacă se anulează în prealabil acest atribut.

Se pot specifica numele fișierelor sub formă generică.

Exemple

— **DELETE * .BAK Q** < *CR* >

Șterge toate fișierele de pe discul 0 cu extensia BAK cerînd pentru fiecare fișier validarea operației din partea utilizatorului.

— **DELETE PROG.ASM P** < *CR* >

LOAD SOURCE DISKETTE, THEN TYPE < CR >
: F0 : PROG.ASM, DELETED
LOAD SYSTEM DISKETTE, THEN TYPE < CR >

Șterge fișierul cu numele PROG.ASM de pe unitatea 0 cerîndu-se schimbarea discului flexibil.

— **DELETE : F1 : PROG1, : F1 : PROG2** < *CR* >

Șterge fișierele PROG1 și PROG2 de pe discul 1.

Comanda ATTRIB

Formatul comenzii este următorul :

ATTRIB < *nume fișier* > < *parametri* > [**Q**]

unde :

< *nume fișier* > este numele fișierului ale cărui atribute se schimbă ;
< *parametri* > reprezintă atributele care se alocă fișierului specificat. Pot lua valorile :

I0 — șterge atributul I (invizibil)
I1 — poziționează atributul I
W0 — șterge atributul W (protejat la scriere)
W1 — poziționează atributul W
F0 — șterge atributul F (format)
F1 — poziționează atributul F
S0 — șterge atributul S (sistem)
S1 — poziționează atributul S
Q — este un parametru prin care se cere validarea operației de schimbare a atributelor de către utilizator.

Comanda **ATTRIB** schimbă atributele fişierelor specificate. Dacă se specifică două valori ale aceluiași atribut, de exemplu I0 și I1, ultimul parametru specificat este cel activ.

Se pot folosi nume generice pentru specificarea fişierelor.

Dialogul de validare a schimbării atributelor, cînd se utilizează parametrul Q, este următorul :

< nume fișier >, MODIFY ATTRIBUTES ?

răspunsul fiind Y („yes") în caz afirmativ și orice caracter în caz negativ.

Atributele curente ale fișierului sînt afișate la consolă.

Exemplu

— **ATTRIB ASM80 · * W1 S1** < *CR* >

Se atribuie tuturor fişierelor cu numele ASM80 şi cu orice extensie atributele **W** (protejat la scriere) și S (sistem).

Comanda RENAME

Formatul comenzii este următorul :

RENAME < *nume fișier vechi* > **TO** < *nume fișier nou* >

unde :

< *nume fișier vechi* > este numele unui fișier care nu are atributul W sau atributul F.

< *nume fișier nou* > este noul nume care se dă fișierului vechi.

Comanda RENAME redenumește numele unui fișier de pe discul flexibil.

Numele vechi și numele nou al fișierului trebuie să fie specificate pe aceeași unitate de disc.

Dacă nume fișier nou există deja pe disc, la consolă apare un mesaj :
< nume fișier nou >, ALREADY EXISTS, DELETE ?

Răspunzînd cu Y („yes") fișierul deja existent se șterge (dacă nu e protejat la scriere) sau nu se întîmplă nimic dacă se răspunde cu orice alt caracter.

Comanda RENAME nu poate fi utilizată pe un sistem cu o singură unitate de disc pentru redenumirea unui fișier de pe un disc flexibil nesistem.

Acest lucru se poate face utilizînd mai întîi o comandă COPY și apoi o comandă DELETE.

Exemplu

— **RENAME PROG.ASM TO PROG.SOU** < *CR* >

Fișierul cu numele PROG.ASM va primi numele PROG.SOU.

6.2.3. Comenzi de conversie de cod

Comenzile de conversie de cod au rolul de a asigura compatibilitatea cu sistemele sau programele ce folosesc fișiere obiect cu format hexazecimal. Conversiile de cod ce se pot realiza sînt :

— din format binar în format hexazecimal — OBJHEX

— din format hexazecimal în format binar — HEXOBJ

Comanda **OBJHEX**

Formatul comenzii este următorul :

OBJHEX *<nume fişier obiect>* **TO** *<nume fişier hexa>*

unde :

<nume fişier obiect> reprezintă numele unui fişier ce conţine cod obiect absolut obţinut în urma executării unor programe de translatare sub sistemul de operare SFDX-18.

<nume fişier hexa> reprezintă numele unui fişier ce va conţine în format hexazecimal (prezentat în paragraful 4.8.3) codul obiect absolut al fişierului obiect specificat.

Comanda OBJHEX converteşte conţinutul unui fişier ce conţine cod obiect absolut, specificat prin *<nume fişier obiect>*, în format hexazecimal şi îi dă numele *<nume fişier hexa>*. Codul obiect absolut specificat este obţinut în urma executării programelor de translatare sub sistemul de operare SFDX-18. În urma execuţiei comenzii pe discul flexibil vor rămîne ambele fişiere, atît în cod obiect absolut cît şi în format hexa.

Adresa de start este luată din fişierul obiect absolut. Codul obiect hexazecimal produs nu conţine o tabelă de simboli.

Un program în format hexa poate fi trecut pe un suport extern şi citit de MON18 prin comanda R.

Exemplu :

— OBJHEX TEST.OBJ TO TEST.HEX *<CR>*

Comanda HEXOBJ

Formatul comenzii este următorul :

HEXOBJ *<nume fişier hexa>* **TO** *<nume fişier obiect>* [**START** (*<adresa>*)]

unde :

<nume fişier hexa> reprezintă numele unui fişier ce conţine cod obiect absolut în format hexazecimal (prezentat în paragraful 4.8.3).

<nume fişier obiect> reprezintă numele unui fişier ce va conţine în format obiect (ce poate fi încărcat de SFDX-18) informaţia specificată de fişierul în format hexazecimal.

<adresă> reprezintă adresa de start (a primei instrucţiuni care se execută) în modulul obiect absolut. Poate fi specificată în hexazecimal, zecimal, octal sau binar (valoarea este postfixată de H pentru hexazecimal, D sau nimic pentru zecimal, Q pentru octal, B pentru binar).

Comanda HEXOBJ converteşte, conţinutul unui fişier ce conţine cod obiect în format hexazecimal, specificat de *<nume fişier hexa>*, în format obiect absolut ce poate fi încărcat de SFDX-18.

Modulul obiect absolut produs de comanda HEXOBJ conţine şi o tabelă de simboli dacă aceştia au fost definiţi.

Se poate include adresa de start (adresa primei instrucţiuni care se execută) în modulul obiect absolut produs de comandă dacă se specifică **START** *(<adresă>)*.

Dacă **START** (*<adresă>*) este omisă, adresa de start este luată din înregistrarea de sfîrşit de fişier hexa, dacă există. Dacă nu este specificată

nici o adresă de start (nici în comandă, nici în înregistrarea de sfîrşit de fişier) atunci ea este presupusă 0. În acest caz, orice încercare de a încărca programul obiect produs în urma executării comenzii, va produce o eroare 15.

Exemplu :

— **HEXOBJ TEST.HEX TO TEST.OBJ START (4000H)** *<CR>*

6.2.4. Comenzile pentru lansarea în execuţie a programelor

Sub controlul sistemului de operare, orice program obiect absolut se lansează în execuţie prin specificarea numelui fişierului ce conţine programul respectiv.

Un program obiect absolut se poate încărca în memorie şi lansa direct în execuţie sau se poate încărca şi lansa în execuţie sub controlul monitorului.

Specificarea programelor sau a comenzilor ce se lansează în execuţie poate fi realizată de la consola sistemului sau de către conţinutul unui fişier de comenzi interpretat de comanda SUBMIT.

Comanda de lansare în execuţie a unui program obiect absolut

Formatul comenzii este următorul :

<nume fişier obiect> [*<parametri>*]

unde :

<nume fişier obiect> reprezintă numele fişierului ce conţine programul obiect absolut ce se lansează în execuţie.

<parametri> reprezintă parametrii necesari programului specificat, pentru controlul execuţiei sale.

Comanda de lansare direct în execuţie a unui program obiect absolut, încarcă programul obiect specificat şi îi transferă controlul. Eventualii parametrii necesari controlului execuţiei sale se pot specifica în cadrul comenzii.

Programul obiect absolut trebuie să fie în cod obiect recunoscut de SFDX18 (nu cod obiect în format hexa), altfel încercarea de lansare în execuţie conduce la o eroare 16. De asemenea adresa de încărcare din cadrul programului trebuie să fie mai mare de 3680H, deoarece o încercare de încărcare peste Zona rezidentă din SFDX18 conduce la o eroare 15.

Exemplu :

Lansarea în execuţie a compilatorului de FORTRAN se face prin comanda :

— **F80** *<CR>*

unde :

F80 reprezintă fişierului ce conţine compilatorul. Lansarea în execuţie a asamblorului ASM80 se face prin comanda :

— **ASM80 TEST.ASM PRINT(:LP:)** *<CR>*

unde :

ASM80 este nume fişierului ce conţine asamblorul în cod obiect absolut.
TEST.ASM este un parametru ce specifică asamblorului, programul pe care urmează să-l analizeze.
PRINT(:LP:) este un parametru ce specifică asamblorului unde să depună fişierul listing rezultat în urma asamblării.

Comanda de lansare în execuție sub controlul monitorului

Formatul comenzii este următorul :

DEBUG [*<nume fișier obiect>* [*<parametri>*]] *<CR>*

unde :

<nume fișier obiect> reprezintă numele fișierului ce conține programul obiect absolut ce se încarcă în memorie.

<parametri> reprezintă eventualii parametrii necesari programului specificat, pentru controlul execuției sale.

Comanda **DEBUG** încarcă în memorie programul obiect absolut specificat de *<nume fișier obiect>*, afișează adresa de lansare în execuție și transferă controlul monitorului. Lansarea în execuție a programului încărcat se face sub controlul monitorului.

Dacă nu se specifică *<numele fișier obiect>* are loc un transfer al controlului de la sistemul de operare SFDX18 la monitorul MON18.

Revenirea în sistemul de operare SFDX18 se poate face astfel :

— introducînd comanda monitor G8 ;
— executînd în programul utilizator un apel de funcție EXIT ;
— executînd în programul utilizator un apel de funcție LOAD cu parametrul 1.

Exemple :

— **DEBUG** *<CR>*
#0008
.

Trece controlul de la sistemul de operare în monitor și așteaptă comenzi monitor.

DEBUG COPY PROG.ASM TO :F1: C *<CR>*
#3680
.

Încarcă programul COPY de pe discul 0 și trece controlul programului monitor. Dacă se dă comanda monitor

.**G** (sau .**G 3680**)

se execută copierea programului PROG.ASM de pe discul 0 pe discul 1.

— **DEBUG TST.OBJ** *<CR>*
#4000
.**G** *4000, 40F3, 4172 <CR>*

Încarcă în memorie programul obiect executabil TST.OBJ, indică adresa de lansare 4000H, după care trece controlul monitorului.

Prin comanda monitor specificată se lansează în execuție programul și se stabilesc două puncte de suspendare temporară a execuției.

Comanda lansare în execuție neinteractivă

Formatul comenzii este următorul :

SUBMIT *<nume fișier comenzi>* [*<extensie>*] [(*parametru* [,*parametru*]*)]

unde :

<nume fișier comenzi> este numele unui fișier sursă ce conține secvența de comenzi

<extensie> reprezintă extensia numelui fișierului ce conține secvența de comenzi. Dacă lipsește se caută implicit extensia **CSD**

<parametru> reprezintă parametrul actual cu care se va înlocui parametrul formal utilizat în secvența de comenzi specificată de fișierul de comenzi.

Comanda **SUBMIT** face ca sistemul de operare să preia comenzile dintr-un fişier ce conţine o secvenţă de comenzi în loc să le preia de la consolă. Înainte de a utiliza comanda **SUBMIT** trebuie să se creeze un fişier ce conţine o secvenţă de comenzi ce pot avea parametri formali.

Lansarea în execuţie a comenzii **SUBMIT** are ca efect prelucrarea comandă cu comandă din fişierul ce conţine secvenţa de comenzi, înlocuirea parametrilor formali cu cei actuali şi lansarea acestora în execuţie.

Comanda poate conţine maxim 10 parametri ce specifică valorile actuale care înlocuiesc parametri formali din secvenţă. Corespondenţa între parametrii actuali şi cei formali se face prin poziţia pe care o ocupa în lista de parametri. Omiterea unui parametru trebuie să fie indicată de o virgulă.

Un parametru este un şir de maximum 31 de caractere alfanumerice sau semne speciale. Dacă un parametru conţine virgulă, blanc sau paranteze el trebuie să fie închis între apostroafe. Dacă în cadrul unui parametru se găseşte un apostrof, acesta se dublează.

Parametrii formali se specifică prin %n unde n este o cifră între 0 şi 9. Pentru ca % să nu fie interpretat ca un parametru formal el trebuie să fie precedat de CTRL/P.

Comanda **SUBMIT** utilizează două fişiere. Unul conţine secvenţa de comenzi cu parametri formali şi este creat de utilizator iar celălalt este creat de comanda cu parametri actuali. Fişierul creat de comandă are acelaşi nume cu cel creat de utilizator dar va avea extensia **CS**.

Comanda **SUBMIT** asignează intrarea de la consola fişierului ce conţine secvenţa de comenzi şi predă controlul sistemului de operare SFDX. Acesta va executa comenzile specificate de secvenţa de comenzi iar la sfîrşit va transfera din nou controlul la consola sistemului. Transferul, mediului de unde se preiau comenzile, între fişierul ce conţine secvenţa de comenzi şi consola sistemului şi invers, se face prin introducerea caracterului de control CTRL/E.

În cadrul fişierului ce conţine secvenţa de comenzi se poate folosi comanda **SUBMIT.** Imbricarea comenzii **SUBMIT** poate avea loc pe maxim 8 nivele.

Dacă în execuţia secvenţei de comenzi controlul revine în SFDX-18 ca urmare a unei erori, fişierul cu extensia CS nu se şterge.

Exemplu.

Presupunem că avem pe discul 1 fişierul ASM.CSD ce conţine următoarea secvenţă de comenzi :

```
ASM %0.%1 PRINT(:%2:%3) %4
CTRL/E
LINK %0.OBJ,SYSTEM.LIB,FPAL.LIB TO %0.LNK
LOCATE %0.LINK PRINT (%0.LOC) MAP CODE (%5)
```

şi că discul cu asamblorul nu conţine programele LINK, LOCATE acestea fiind pe un alt disc.

Schimbarea discului 0 se poate face utilizînd caracterul de control CTRL/E care va comuta controlul din secvenţa de comenzi de pe disc la consolă. După schimbarea discului la consolă se va tasta CTRL/E care va trece din nou controlul în secvenţa de comenzi de pe disc.

Execuţia comenzii :

— **SUBMIT :F1:ASM (:F1:TEST.ASM, LP, ,MACROFILE, 4000H)** <CR>

este echivalentă cu secvenţa de comenzi

— ASM80 :F1:TEST.ASM PRINT(:LP:) MACROFILE

CTRL/E CTRL/E (se trece controlul la consolă şi înapoi, timp în care se poate schimba discul 0)
— LINK :F1:TEST.OBJ, SYSTEM.LIB,FPAL.LIB TO :F1:TEST.LNK
— LOCATE :F1:TEST.LNK PRINT(:F1:TEST.LOC) MAP CODE (4000H)
— :F0: SUBMIT RESTORE :F1:TEST.CS (:TI:)

6.2.5. Comandă de întreţinere disc

Comanda de întreţinere disc are rolul de a verifica dacă zonele neocupate de pe disc răspund corect la operaţiile de citire-scriere. Comanda verifică, pistă cu pistă, dacă operaţiile de citire-scriere se desfăşoară normal. În cazul în care apar erori, sînt marcate ca ocupate pistele defecte.

Comanda VDISK

Formatul comenzii este următorul :

VDISK [<*parametru*>]

unde :

<*parametru*> — are una din valorile 0, 1, 2, 3 şi specifică unitatea de disc implicată în operaţia de verificare. Dacă parametrul este omis se consideră implicit unitatea 0.

Comanda verifică pistă cu pistă, Zonele neocupate de pe unitatea specificată. Dacă operaţiile de citire/scriere se desfăşoară normal se trece la pista următoare. În caz contrar se marchează în fişierul SFDX.MAP ocuparea acestor blocuri şi se afişează la consolă numărul pistelor defecte. Marcarea are ca rol eliminarea acestor blocuri din operaţiile curente de alocare la fişierele ce urmează să fie create pe disc. Dacă se specifică unitatea 0, are loc un dialog pentru schimbarea discului flexibil. Dialogul are următoarea structură :

LOAD DISKETTE, THEN TYPE (CR)
LOAD SYSTEM DISKETTE, THEN TYPE (CR)

Exemple.

—VDISK <CR>
LOAD DISKETTE, THEN TYPE (CR)
NO DEFECTIVE TRACKS
LOAD SYSTEM DISKETTE, THEN TYPE (CR)

După încărcarea în unitatea 0 a discului ce urmează să se analizeze se introduce <CR>. Ca urmare a analizei a rezultat că nici o pistă nu este defectă.

—VDISK 1 <CR>
DEFECTIVE TRACKS : 53 69 71 72
104 BLOCKS RESERVED

În urma analizei discului din unitatea 1 a rezultat că pistele cu numărul de ordine 53 69 71 72 sînt defecte şi nu vor fi utilizate în exploatarea sistemului.

6.2.6. Comenzi pentru lucrul cu module program

Elaborarea programelor complexe, sub formă liniară într-un singur bloc, necesită foarte mare efort din partea utilizatorului atît în faza de translatare cît şi în cea de testare. Programele se pot elabora cu un efort mai mic dacă se utilizează module mai mici şi mai simple care interacţionează între ele.

Sistemul de operare SFDX-18 conţine o serie de comenzi care manipulează module de program în scopul de a construi din ele un program complex. Aceste comenzi se referă la :

— combinarea mai multor module în unul singur — **LINK**

— alocarea de adrese absolute de memorie modulelor relocatabile — **LOCATE**

— întreţinerea/manipularea bibliotecilor de programe — **LIB**

Comanda LINK

Formatul comenzii este următorul :

LINK *<listă de intrare>* **TO** *<fişier de ieşire>* [*<parametri>*] <CR>

unde :

<lista de intrare> cuprinde :

— nume de fişiere sub forma *<nume fişier>*, *<nume fişier>*, ... care conţin module obiect generate de translatoare sau de comenzi **LINK**, **LOCATE** anterioare, care vor fi legate în fişierul de ieşire ;

— referiri la simboli externi absoluţi sub forma **PUBLICS** (*<nume fişier absolut>*, *<nume fişier absolut>*, ...) fără ca fişierele absolute să fie legate în fişierul de ieşire ;

— nume de biblioteci cu module obiect sub forma *<nume fişier bibliotecă>* [(*<nume modul>*), ...], a căror module care satisfac referinţe nerezolvate vor fi legate în fişierul de ieşire,

<fişier de ieşire> specifică fişierul care va conţine modulul obiect relocatabil rezultat din legarea modulelor de intrare

<parametri> — reprezintă parametrii de control ai execuţiei comenzii **LINK.** Pot lua valorile :

MAP — produce o hartă a alocărilor de memorie (de legături pentru fişierul de ieşire) ;

NAME (*<nume modul>*) — specifică numele care va fi atribuit modulului de ieşire. Implicit, numele este format din numele utilizat pentru fişierul de ieşire prin eliminarea extensiei

PRINT (*<nume fişier>*) — specifică numele fişierului în care se va scrie harta alocărilor de memorie. Dacă acest parametru este omis, scrierea se face în mod implicit la consola sistemului.

Comanda **LINK** combină module obiect din mai multe fişiere de intrare într-un singur modul de ieşire. În cadrul procesului de combinare are loc o ajustare a adreselor relative ale modulelor de intrare. De asemenea, comanda **LINK** caută în biblioteci şi leagă modulele din biblioteci care rezolvă referinţele externe, la modulele specificate în lista de intrare şi le include în fişierul de ieşire.

Comanda **LINK** poate să se continue pe mai multe linii (rînduri). Pentru aceasta trebuie introdus caracterul & înainte de sfîrşitul liniei (CR), dar nu în interiorul unui nume de fişier sau parametru.

De subliniat faptul că în cadrul execuţiei comenzii LINK se utilizează un fişier temporar LINK.TMP pe discul pe care se va depune fişierul de ieşire.

Harta alocărilor de memorie conţine următoarele informaţii :
— o linie care specifică faptul că urmează mesaje de LINK ;
— lungimea segmentelor relocatabile din modulul de ieşire ;
— adresele absolute în modulul de ieşire ;
— numele modulelor de intrare ;
— numele referinţelor externe nerezolvate ;
— mesajele erorilor nefatale ;
— tipul relocatării :

B relocatabil la nivel de octet ;
P relocatabil la nivel de pagină ;
I relocatabil în pagină ;
A absolut.

Comanda **LINK** combină modulele din lista de intrare, prin unirea segmentelor de cod din modulele de intrare, într-un singur segment de cod, segmentele de date într-un singur segment de date, segmentele de stivă într-un singur segment de stivă.

Ordinea de legare a modulelor urmăreşte ordinea în care apar modulele în lista de intrare. Dacă anumite module din lista de intrare au referinţe la un modul dintr-un fişier bibliotecă, biblioteca trebuie specificată în lista de intrare după modulele care se referă la ea.

Exemplu.

```
SFDX-18 OBJECT LINKER V3.0 INVOKED BY :
— LINK :F1:P80.OBJ.&
**          :F0:SYSTEM LIB, &
**          :F1:PLM80.LIB TO :F1: P80.LNK MAP PRINT (:LP:)
LINK MAP OF MODULE P80
WRITTEN TO FILE :F1: P80.LNK
MODULE IS A MAIN MODULE
SEGMENT INFORMATION :
START STOP LENGTH REL NAME
                 146H B CODE
      0FH B DATA
      8H B STACK
INPUT MODULES INCLUDED :
:F1:P80.OBJ(SFDX80)
:F0:SYSTEM.LIB(IODEF)
:F0:SYSTEM.LIB(SFDX)
:F1:PLM80.LIB(CP0014)
```

Comanda LOCATE

Formatul comenzii este următorul :

LOCATE <*nume fișier intrare*> [**TO**<*nume fișier ieșire*>] [<*parametri*>] <CR>

unde :

<*nume fișier intrare*> — reprezintă numele fișierului obiect care conține codul relocatabil ;

<*nume fișier ieșire*> — reprezintă numele fișierului obiect care va conține codul absolut după execuția comenzii. Dacă este omis, fișierul de ieșire va avea același nume cu cel de intrare fără extensie ;

<*parametru*> — reprezintă parametrii de control ai execuției comenzii **LOCATE**. Pot lua valorile :

MAP — produce o hartă a alocărilor de memorie ce va fi depusă în fișierul de listare ;

PRINT (<*nume fișier*>) — specifică fișierul care va conține listingul generat pe parcursul execuției comenzii. Dacă nu este specificat, listarea se va face la consolă (:CO:) ;

SYMBOLS — va introduce în fișierul de listare o listă cu simbolii definiți în modulele de program, cu adresele absolute la care se vor găsi acești simboli în timpul execuției modulelor de program respective. Această listă se poate obține pentru modulele rezultate în urma compilării PLM80, numai dacă la compilare s-a folosit parametrul **DEBUG**.

LINES — specifică includerea unei liste cu numerele de linii și numele modulelor de intrare care urmează să fie incluse în tabela de simboli. Se va genera o listă a numerelor instrucțiunilor, cu adresele absolute de memorie la care se va găsi codul obiect pentru aceste instrucțiuni.

PUBLICS — în fișierul de listare se va include lista simbolurilor declarați **PUBLIC** cu adresele absolute de memorie la care se vor afla acești simboli în timpul execuției programului.

PURGE — condensează dimensiunea codului, eliminînd din fișierul de ieșire numerele de linii, simbolii locali, numele de module și numele simbolilor **PUBLIC**. Acest parametru este utilizat în momentul în care un modul este complet depanat.

ORDER (<*secvență segmente*>) — definește ordinea în care sînt alocate în memorie segmentele de cod, stivă, date. Lista de segmente se separă prin spațiu. Dacă lista de segmente este parțială, segmentele specificate sînt alocate primele, în ordinea din secvență iar cele nespecificate urmează secvența prestabilită. Secvența prestabilită este următoarea :

CODE — segmentul de cod
STACK — segmentul de stivă
/Zona COMMON/ — segment de comun pentru FORT80
DATA — segmentul de date
MEMORY — segmentul de memorie disponibilă

Primul segment este alocat implicit la adresa 3680H, alocîndu-se 13 buffere de intrare/ieșire.

CODE (<*adresă*>) — fixează adresa de memorie la care se va depune segmentul care conține codul obiect.

DATA (<*adresă*>) — fixează adresa de memorie la care se va depune segmentul care conține datele.

STACK (<*adresă*>) — fixează adresa de memorie la care se va depune segmentul care conține stiva.

MEMORY (<*adresă*>) — fixează adresa de memorie la care se va depune segmentul care conține zona de memorie disponibilă.

NAME (<*nume*>) — specifică numele modulului de ieșire. Dacă acest parametru nu este utilizat se va considera implicit numele fișierului de intrare, fără extensie.

RESTART0 — este un parametru care va depune în locațiile 0, 1, 2 din memorie o instrucțiune JMP ADR. Adresa ADR este adresa de început a programului, luată din modulul de intrare sau din parametrul START.
Un modul absolut pregătit cu **RESTART0** nu poate fi încărcat pentru execuție cu SFDX-18.

START (*<adresă>*) — specifică adresa primei instrucțiuni care se execută din segmentul de cod. Această adresă înlocuiește adresa de start specificată în modulul de intrare.

STACKSIZE (*<valoare>*) — specifică dimensiunea segmentului de stivă (în octeți). Această valoare este cea considerată în cazul în care există o altă valoare calculată pentru dimensiunea stivei în modulul de intrare. De notat faptul că în regim de depanare sînt necesari 12 octeți adiționali pentru stiva utilizator pe lîngă cei calculați de comanda **LINK.** Comanda **LOCATE** adaugă acești 12 octeți dacă controlul **STACKSIZE** nu este specificat.

COLUMNS (*<număr>*) — este un parametru care specifică coloana din care se tipărește tabela de simboli în fișierul listing. (*<Număr*)> poate lua valorile 1, 2, 3.
Comanda **LOCATE** acționează asupra fișierului de intrare care conține un modul obiect relocatabil și produce un fișier modul obiect în care adresele relative sînt fixate la locații absolute. Fișierul obiect absolut obținut este direct executabil.
Dacă comanda **LOCATE** este mai lungă decît o linie de la consolă ea se poate continua pe mai multe linii, introducînd caracterul & înaintea fiecărui <CR>.
Comanda **LOCATE** utilizează un fișier temporar LOCATE. TMP pe discul unde se depune fișierul de ieșire.
Modulele segment sînt alocate secvențial în ordinea implicită CODE STACK /COMMON/ DATA MEMORY sau în ordinea stabilită prin parametrul **ORDER** sau prin parametrii **CODE** (*<adresă>*), **STACK** (*<adresă>*), **DATA** (*<adresă>*), **MEMORY** (*<adresă>*).

Segmentele relocatabile la nivel de octet (B) sînt locate la prima adresă disponibilă, cele relocatabile la nivel de pagină (P) sînt locate la prima adresă care rezultă la multipli de pagină (100H) iar cele relocatabile în pagină (I) sînt locate la prima adresă disponibilă astfel încît segmentul să fie conținut în totalitate într-o pagină.

Exemplu.

```
SFDX-18 OBJECT LOCATER V3.0 INVOKED BY:
— LOCATE:F1:P80.LNK TO:F1:P80 MAP SYMBOLS PRINT(:LP:) COLUMNS(2)
SYMBOL TABLE OF MODULE P80
READ FROM FILE :F1:P80.LNK
WRITTEN TO FILE :F1:P80
VALUE TYPE SYMBOL                     VALUE TYPE SYMBOL
           MOD SFDX80
 37E9H   SYM  MEMORY                   3680H   SYM  SPEC
 4100H   SYM  COMAREA                  37DAH   SYM  I
 37DBH   SYN  J                        37DCH   SYM  POSITION
 37DDH   SYM  OBLOCK                   37E7H   SYM  IBLOCKADR
```

```
3796H   SYM  INT6                        36A5H   SYM  WAIT
378DH   SYM  ANSWERBACK
MEMORY MAP OF MODULE P80
READ FROM FILE :F1:P80.LNK
WRITTEN TO FILE :F1:P80
MODULE START ADDRESS 369AH
START STOP LENGTH REL NAME
3680H  3705H  146H   B CODE
3706H  37D9H  14H    B STACK
37DAH  37E8H  FH     B DATA
37E9H  DE76H  A68EH  B MEMORY
```

Comanda LIB

Formatul comenzii este următorul :

LIB

Comanda **LIB** este interactivă, operațiile efectuate sînt controlate prin comenzi introduse de la consolă după ce sistemul a tipărit la consolă *.

Funcțiile îndeplinite de comanda **LIB** sînt următoarele :

— crearea unui fișier bibliotecă — **CREATE.**
— adăugarea modulelor la fișierul bibliotecă — **ADD.**
— ștergerea modulelor din fișierul bibliotecă — **DELETE.**
— listarea numelor modulelor bibliotecă și a simbolilor lor publici **PUBLICS — LIST.**
— întoarcerea în sistemul de operare — **EXIT.**

Bibliotecile pot fi utilizate ca module de intrare pentru comanda **LINK.**

O comandă din cadrul programului **LIB** poate fi continuată pe mai multe linii introducînd caracterul & înainte de <CR>.

Funcția CREATE

Creează un fișier bibliotecă gol. Sintaxa de descriere a funcției este :

CREATE <*nume fișier*> <CR>.

unde : <*nume fișier*> — specifică numele care se atribuie noului fișier bibliotecă.

Dacă există deja un fișier cu numele specificat se afișează o eroare și se așteaptă introducerea unei noi funcții.

Funcția ADD

Sintaxa funcției este următoarea :

ADD <*fișier sursă*> [(<*nume modul*>,...)] [,...] **TO** <*nume bibliotecă*> <CR>.

unde :

<*fișier sursă*> poate fi numele unui fișier bibliotecă sau numele unui fișier ce conține un modul obiect.

<*nume modul*> se specifică numai dacă <*fișier sursă*> este un fișier bibliotecă. În acest caz numai modulele obiect specificate, din cadrul bibliotecii sînt adăugate la ieșire în nume bibliotecă.

<*nume bibliotecă*> este fișierul bibliotecă la care se adaugă fișierele sau modulele specificate ca intrare.

Funcția **ADD** adaugă module obiect la fișierul bibliotecă.

Funcţia DELETE

Sintaxa funcţiei este următoarea :

DELETE <*fişier bibliotecă*> (<*nume modul*>, . . .)

unde :

<*fişier bibliotecă*> specifică fişierul bibliotecă din care se şterg module.

<*nume modul*> specifică numele modulului obiect care se şterg din fişierul bibliotecă.

Funcţia **DELETE** şterge module din fişierul bibliotecă.

Funcţia LIST

Sintaxa funcţiei este următoarea :

LIST <*fişier bibliotecă*> [(<*nume modul*>, . . .)] [, . . .] [**TO** <*fişier listing*>] [**PUBLICS**] <CR>.

unde :

<*fişier bibliotecă*> este numele fişierului bibliotecă ale cărui module sînt listate.

<*nume modul*> este numele modulului despre care sînt listate informaţii. Dacă nume modul este omis se vor lista toate modulele din fişierul bibliotecă.

<*fişier listing*> este numele fişierului în care se va depune informaţii despre fişierul bibliotecă.

Dacă <*fişier listing*> este omis se face listarea la consolă (:CO:).

PUBLICS — este un parametru care specifică, ca numele variabilelor globale (publice) din fiecare modul să fie listate. Dacă acest parametru este omis sînt listate numai numele modulelor.

Funcţia **LIST** listează numele modulelor fişierului bibliotecă specificat, cu listarea eventualilor simboli PUBLICI din fiecare modul listat.

Funcţia EXIT

Sintaxa funcţiei este următoarea :

— **EXIT** <CR>.

Comanda **EXIT** întoarce controlul în sistemul de operare SFDX-18.

Exemplu.

```
— LIB
* LIST : F1 : SYSTEM.LIB <CR>
  :F1:SYSTEM.LIB
  CI
  CO
  CSTS
  IOCHK
  IODEF
  IOSET
  LO
  MEMCK
  PO
  RI
  SFDX
  ATTRIB
  CLOSE
  CONSOL
  DELETE
```

EXIT
LOAD
OPEN
READ
RENAME
RESCAN
SEEK
SPATH
WRITE
ERROR
WHOCON

6.3. Utilizarea rutinelor de sistem

Pentru a putea utiliza rutine din sistemul de operare, este necesar să se ţină seama de faptul că un program utilizator nu se poate încărca, cu comenzi SFDX, peste zona în care este rezident sistemul de operare. Este important să se delimiteze zonele de memorie utilizator de zonele sistem.

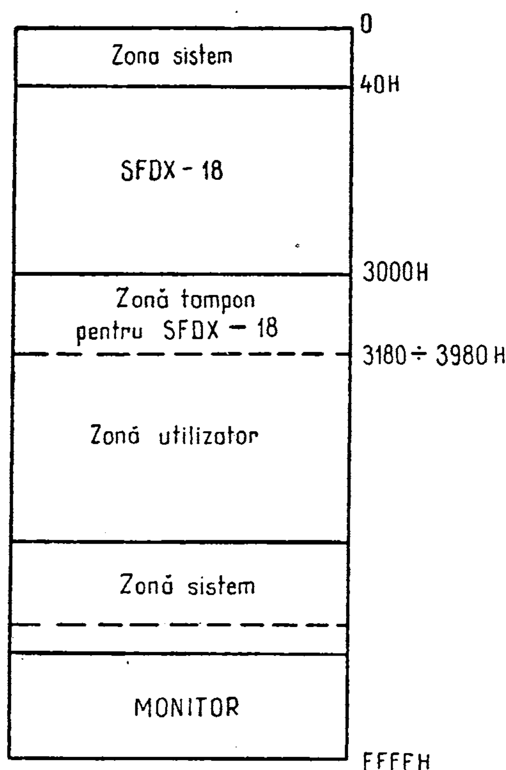


Fig. 6.7. Repartizarea zonelor sistem şi a zonelor utilizator.

Repartizarea zonelor utilizator şi a zonelor sistem este prezentată în figura 6.7.

Zona de la adresele 0÷40H este rezervată sistemului şi conţine informaţie necesară monitorului, adresele de legătură pentru întreruperile de nivel 0÷2 necesare sistemului de operare şi adresele de legătură pentru întreruperile de nivel 3÷7 controlate de utilizator.

Zona sistem de la sfîrşitul memoriei este necesară programului monitor şi a fost descrisă în detaliu în capitolul 4.

Dimensiunea zonei ocupată de monitor este funcţie de versiunea acestuia.

Sistemul de operare SFDX-18 ocupă zona de la 40H pînă la 3000H. SFDX-18 are nevoie de o zonă în care să se păstreze zonele tampon de intrare/ieşire necesare în exploatarea fişierelor. Zona tampon are o dimensiune între 180H şi 980H, funcţie de numărul de buffere de intrare/ieşire necesare. Lungimea unui buffer de intrare/ieşire este de 128 (80H)

octeți. Sistemul de operare are nevoie de minimum 3 buffere de I/E, alocate consolei. Consola are, în permanență, deschise fișiere asociate intrării, ieșirii și editării.

Adresa de început a programului utilizator se stabilește în funcție de zona tampon necesară pentru bufferele de I/E. Zona tampon se calculează ținîndu-se seama de următoarele reguli :

— fiecare fișier deschis pe un disc flexibil necesită două buffere ;

— un apel la rutinele sistemului de operare ce face acces la fișierul director necesită două buffere pe perioada apelului ;

— o asignare a unui fișier pe disc ca și intrare de la consolă sau ieșire la consolă necesită trei buffere pentru fișierul asociat intrării de la consolă și două buffere pentru fișierul asociat ieșirii la consolă ;

— un fișier pentru editat o linie de la :CI: necesită un buffer.

Ținînd seama de regulile prezentate, zona tampon pentru buffere de I/E se calculează astfel :

Zona tampon=80H*N.

unde : N reprezintă numărul de buffere de I/E necesare.

Astfel pentru un program care deschide un fișier pe discul flexibil și se dorește să fie apelat dintr-un fișier SUBMIT unde consola de ieșire este asignată unui fișier pe disc sînt necesare 9 buffere de intrare/ieșire. Adresa sa de încărcare poate fi minimum 3480H=3000H+80H×9.

Dacă se dorește scrierea unui program independent de tipul perifericelor utilizate și independent de modul de apelare, trebuie prevăzută o zonă tampon ce conține numărul maxim de buffere de I/E ce ar putea fi necesare. Numărul maxim de buffere de I/E admis este 19. În acest caz programul trebuie să înceapă la o adresă mai mare de 3980H.

Rutinele de sistem ce pot fi apelate dintr-un program utilizator realizează următoarele acțiuni :

— intrare/ieșire de la echipamentele standard conectate la M18/M118 inclusiv discul flexibil ;

— întreținerea fișierului director a discurilor flexibile ;

— asignarea consolei și transmiterea mesajelor de eroare ;

— încărcarea programelor pentru execuție.

Majoritatea apelurilor rutinelor de sistem au nume și funcții similare comenzilor SFDX-18 prezentate anterior. Apelurile la rutinele de sistem pot fi făcute din limbaj de asamblare sau limbaj PL/M.

Interfața între programele scrise în limbaj de asamblare și rutinele de sistem este realizată prin apelul rutinei SFDX și specificarea a doi parametri.

Primul parametru este un număr ce specifică funcția (rutina de sistem) iar al doilea parametru este adresa unui bloc de control ce conține informația necesară funcției apelate. Primul parametru este transferat prin registrul C iar adresa blocului de control este transmisă prin registrele D, E.

Rutina SFDX este o rutină din SYSTEM.LIB, deci trebuie declarată externă în programele utilizator ce vor fi legate prin comanda LINK, cu biblioteca sistemului. În limbajul PL/M, interfața cu rutinele de sistem este realizată cu ajutorul procedurilor din SYSTEM.LIB. Programul PL/M trebuie să conțină declarațiile procedurilor externe apelate din

SYSTEM.LIB astfel încît procedurile corespunzătoare să fie incluse cu ajutorul comenzii LINK. Apelul rutinelor de sistem folosește stiva utilizatorului ceea ce implică dimensionarea corespunzătoare a acesteia. În urma apelurilor rutinelor de sistem, conținutul registrelor generale este modificat.

Numerele ce identifică apelurile rutinelor de sistem sînt următoarele :

OPEN	0
CLOSE	1
DELETE	2
READ	3
WRITE	4
SEEK	5
LOAD	6
RENAME	7
CONSOL	8
EXIT	9
ATTRIB	10
RESCAN	11
ERROR	12
WHOCON	13
SPATH	14

Fiecărui fișier îi sînt asociați doi indicatori și anume :
LENGTH — ce conține numărul de octeți din fișier ;
MARKER — ce indică numărul de octeți deja citiți sau scriși în fișier.

6.3.1. OPEN — Deschide un fișier pentru operații de intrare/ieșire

Rutina **OPEN** inițializează tabelele SFDX și alocă bufferele necesare pentru operația de intrare/ieșire la fișierul specificat. Dacă fișierul specificat este :PP: atunci este perforat un cap de bandă de 30 cm.

Blocul de control al funcției conține 5 parametri, fig. 6.8. care au următoarea semnificație :

AFTN — reprezintă adresa unei zone de 2 octeți în care SFDX va depune numărul fișierului activ (numărul logic) ce se va deschide (dacă deschiderea s-a realizat corect)

FILE — adresa de început a zonei de memorie care conține numele fișierului sub formă de caractere ASCII. Numele fișierului trebuie să fie urmat de un caracter special, diferit de două puncte (:) sau punct (.). De obicei se termină cu spațiu ( ) sau (CR)

ACCES — specifică modul de acces la fișierul deschis
=1 pentru citire (READ)
=2 pentru scriere (WRITE)
=3 pentru punere la zi (READ și WRITE)

MODE — numărul logic al fișierului ecou, dacă fișierul este deschis pentru editat linie cu linie. Fișierul ecou trebuie deschis în prealabil. Parametrul AFTN al fișierului ecou este depus în cel mai puțin semnificativ octet al cîmpului de 2 octeți
=0 nu este realizată editarea
≠0 este realizată editarea

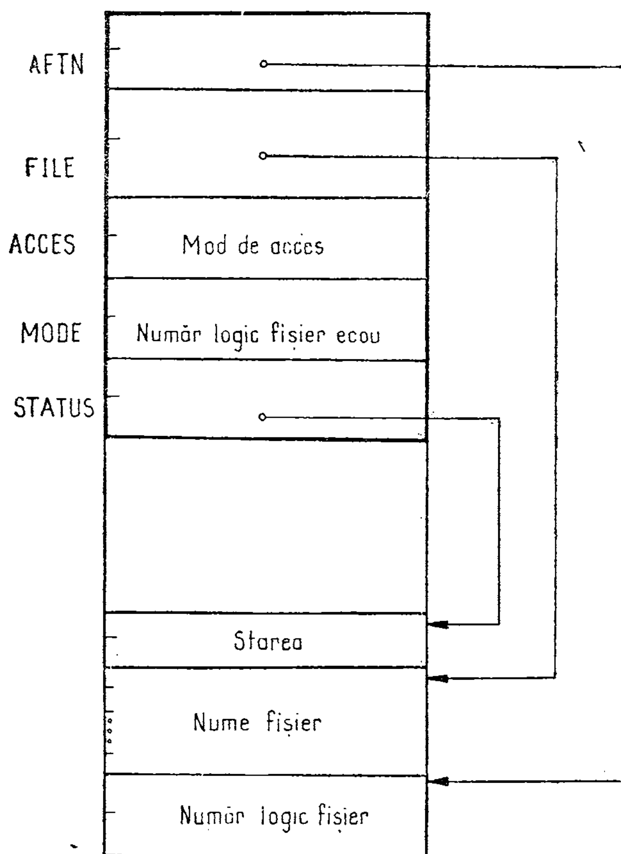


Fig. 6.8. Structura blocului de control pentru funcția OPEN.

STATUS — specifică adresa unei celule de memorie unde se depune numărul erorii nefatale ce a apărut în timpul execuției rutinei OPEN sau 0 dacă totul a decurs normal.

Sînt necesare cîteva precizări suplimentare și anume :

— Utilizatorul poate avea maxim 6 fișiere deschise la un moment dat, exclusiv fișierele :CI: și :CO:

— Fișierele :CI: și :CO: sînt în permanență deschise și au asociate ca număr logic (AFTN) valoarea 1 respectiv valoarea 0.

— Pentru a specifica, la parametrul MODE, numărul fișierului :CO: (care are AFTN=0) trebuie depusă o valoare diferită de 0 în octetul cel mai semnificativ și 0 în octetul cel mai puțin semnificativ. De exemplu se poate folosi valoarea 0FF00H pentru a specifica :CO:

— Dacă un fișier este deschis pentru intrare (READ) cei doi indicatori MARKER și LENGTH vor fi :

MARKER=0
LENGTH=nemodificat

— Dacă un fişier este deschis pentru ieşire (WRITE) cei doi indicatori vor fi :
MARKER=0
LENGTH=0

— Dacă un fişier este deschis pentru actualizare (READ, WRITE) cei doi indicatori vor fi :
MARKER=0
LENGTH=neschimbat (dacă fişierul deschis deja există)

— Specificarea unui fişier ce nu există pe disc pentru operaţia de ieşire (WRITE) va produce crearea unui nou fişier cu numele respectiv şi toate atributele vor fi puse la 0.

— Specificarea unui fişier de intrare inexistent, a unui fişier protejat la scriere pentru operaţiile de actualizare sau scriere va produce un mesaj de eroare.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
         EXTRN   SFDX             ; LEGĂTURA CU PUNCTUL DE IN-
                                  ; TRARE ÎN SFDX
OPEN     EQU     0                ; IDENTIFICATORUL RUTINEI OPEN
         MVI     C, OPEN          ; ÎN C NUMĂR IDENTIFICARE RUTINĂ
         LXI     D, OBLK          ; ÎN D, E ADRESA BLOCULUI DE PA-
                                  ; RAMETRI
         CALL    SFDX             ; APELARE RUTINĂ
         LDA     OSTAT            ; TEST INDICATOR EROARE
         ORA     A                ;
         JNZ     ERR              ; SALT LA RUTINA DE TRATARE
         .                        ; A ERORII, DACĂ ESTE CAZUL
         .
         .
OBLK :                            ; BLOCUL DE PARAMETRI PENTRU
                                  ; OPEN
         DW      OAFT             ; ADRESA SPRE AFTN
         DW      OFILE            ; ADRESA SPRE NUME FIŞIER
ACCES :  DW      1                ; MOD ACCES=1, READ
ECHO :   DW      0                ; FĂRĂ ECOU
         DW      OSTAT            ; ADRESA INDICATOR EROARE
OAFT :   DS      2                ; CONŢINE AFTN (NUMĂR FIŞIER
                                  ; ACTIV)
OSTAT :  DS      2                ; CONŢINE STAREA (EVENTUAL NR
                                  ; ERORII)
OFILE :  DB      ':F1:PROG.ASM '  ; NUMELE FIŞIERULUI DESCHIS
```

b) În limbajul PL/M

```
OPEN :
PROCEDURE (AFTNPTR, FILE, ACCES, MODE, STATUS), EXTERNAL ;
DECLARE (AFTNPTR, FILE, ACCES, MODE, STATUS) ADDRESS ;
END OPEN ;
.
.
.
DECLARE AFT$IN ADDRESS ;
DECLARE BUFFER (128) BYTE ;
DECLARE STAT ADDRESS ;
.
.
.
CALL OPEN (.AFT$IN, .BUFFER, 1, 0, .STAT) ;
```

6.3.2. CLOSE — Termină operaţiile de intrare/ieşire la un fişier

Rutina **CLOSE** eliberează bufferele alocate fişierului la deschiderea sa. Dacă fişierul închis este perforatorul de bandă (:PP:) va fi perforat un sfîrşit de bandă de 30 cm.

Blocul de control al funcţiei conţine 2 parametri, fig. 6.9, care au următoarea semnificaţie :

AFTN — conţine numărul logic al fişierului ce urmează să fie închis, număr ce a fost fixat la deschidere de rutina OPEN.

STATUS — specifică adresa unei locaţii de memorie unde se va depune eventualul cod de eroare nefatală. Dacă totul a decurs normal la această adresă se va găsi 0.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
         EXTRN   SFDX         ; LEGATURA CU PUNCTUL DE IN-
                              ; TRARE ÎN SFDX
CLOSE    EQU     1            ; IDENTIFICATORUL RUTINEI CLOSE
         MVI     C,CLOSE      ; IN C NUMARUL DE IDENTIFICARE
         LXI     D,CBLK       ; RUTINA IN D,E ADRESA BLOCULUI
                              ; DE PARAMETRI
         CALL    SFDX         ; APELARE RUTINA
         LDA     CSTAT        ; TEST INDICATOR EROARE
         ORA     A            ;
         JNZ     ERR          ; SALT LA RUTINA DE TRATARE
         .                    ; A ERORII, DACA ESTE CAZUL
         .
         .
CBLK:                         ; BLOCUL DE PARAMETRI PENTRU
                              ; CLOSE
CAFT:    DS      2            ; NUMARUL LOGIC AL FISIERULUI
         DW      CSTAT        ; ADRESA INDICATOR EROARE
CSTAT:   DS      2            ; CONTINE STAREA (EVENTUAL NR
                              ; ERORII)
```

b) În limbajul PL/M

```
CLOSE:
    PROCEDURE (AFTN, STATUS) EXTERNAL ;
    DECLARE (AFTN, STATUS) ADDRESS ;
    END CLOSE ;
    .
    .
    .
    DECLARE AFT$IN ADDRESS ;
    DECLARE STAT$IN ADDRESS ;
    .
    .
    .
    CALL CLOSE (AFT$IN,STAT$IN) ;
    .
    .
    .
```

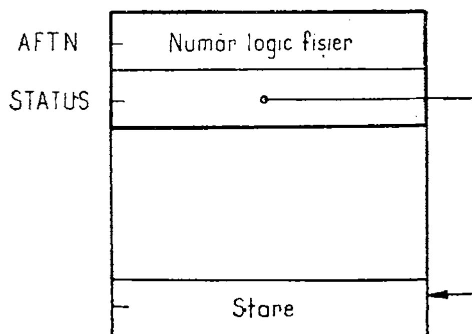


Fig. 6.9. Structura blocului de control pentru funcţia CLOSE.

6.3.3. DELETE — Şterge un fişier din fişierul director

Apelul rutinei DELETE eliberează spaţiul alocat fişierului ce urmează să se şteargă. Blocul de control al funcţiei conţine 2 parametri, fig. 6.10, care au următoarea semnificaţie :

FILE : — adresa de început a zonei de memorie ce conţine numele fişierului sub formă de şir de caractere ASCII. Numele fişierului trebuie urmat de un caracter special diferit de două puncte (:) sau punct (.). De obicei se termină cu spaţiu (ъ) sau (CR).

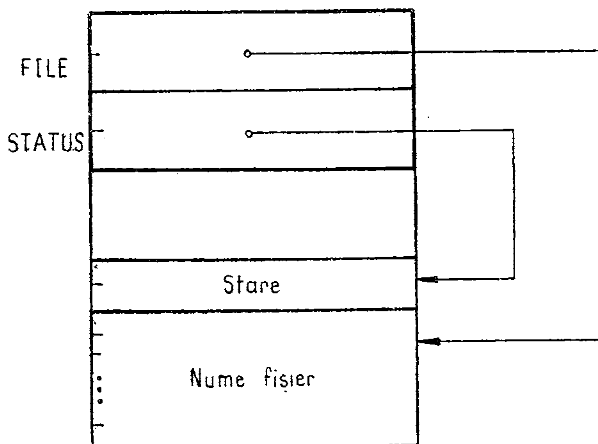


Fig. 6.10. Structura blocului de control pentru funcţia DELETE.

STATUS — specifică adresa unei locaţii de memorie unde se va depune eventualul cod de eroare nefatală, rezultată în urma apelului rutinei. Dacă totul decurge normal la această adresă se va găsi 0. Apelul rutinei se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX          ; LEGATURA CU PUNCTUL DE IN-
                              ; TRARE ÎN SFDX
DELETE  EQU     2             ; IDENTIFICATORUL RUTINEI DELETE
        MVI     C,DELETE      ; ÎN C NUMARUL DE IDENTIFICARE
                              ; RUTINĂ
        LXI     D,DBLOK       ; ÎN D,E ADRESA BLOCULUI DE PA-
                              ; RAMETRI
        CALL    SFDX          ; APELARE RUTINĂ
        LDA     DSTAT         ; TEST INDICATOR EROARE
        ORA     A             ;
        JNZ     ERR           ; SALT LA RUTINA DE TRATARE A
                              ; ERORII
.                             ; DACĂ ESTE CAZUL
.
.
DBLK:                         ; BLOCUL DE PARAMETRI PENTRU
                              ; DELETE
        DW      DFILE         ; ADRESA SPRE NUMELE FISIERULUI
        DW      DSTAT         ; ADRESA SPRE INDICATORUL DE
                              ; EROARE
DSTAT:  DS      2             ; CONTINE STAREA
DFILE:  DB      'PROG.TST' '  ; NUMELE FISIERULUI CE SE ŞTERGE
```

b) În limbaj PL/M

```
DELETE:
     PROCEDURE (FILE, STATUS) EXTERNAL ;
       DECLARE (FILE, STATUS) ADDRESS ;
     END DELETE ;
     .
     .
     .
     DECLARE FILENAME (20) BYTE ;
     DECLARE STAT$IN ADDRESS ;
     .
     .
     .
     CALL DELETE (.FILENAME,.STAT$IN) ;
     .
     .
     .
```

6.3.4. READ — Transferă date dintr-un fişier în memorie

Rutina **READ** transferă date dintr-un fişier deschis, într-o zonă de memorie specificată.

Blocul de control al funcţiei conţine 5 parametri, fig. 6.11, care au următoarea semnificaţie :

AFTN — conţine numărul logic al fişierului deschis pentru citire sau actualizare, număr ce a fost fixat la deschidere de rutina OPEN.

BUFFER — conţine adresa zonei de memorie în care se vor transfera datele din fişierul deschis anterior de apelul unei rutine OPEN.

COUNT — conţine numărul de octeţi ce urmează să fie transferaţi din fişier în zona de memorie specificată. Zona de memorie trebuie să fie egală sau mai mare decît contorul deoarece, în caz contrar,

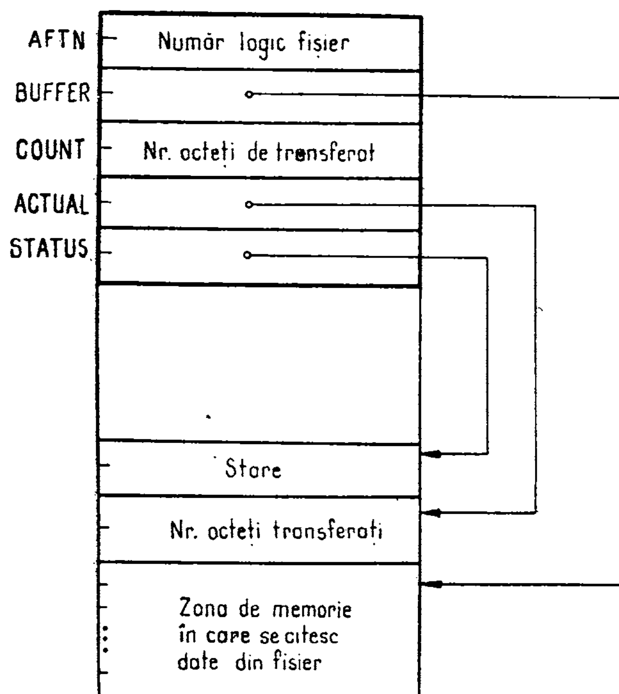


Fig. 6.11. Structura blocului de control pentru funcția READ.

zona ce urmează bufferului va fi utilizată și poate distruge informația utilă.

ACTUAL — conține adresa unei celule de memorie în care SFDX va memora numărul actual de octeți transferați

STATUS — conține adresa unei celule de memorie în care SFDX va memora numărul eventualei erori nefatale rezultată în urma apelării rutinei READ. Dacă totul decurge normal la această adresă se găsește 0.

De subliniat faptul că numărul actual de octeți transferați nu este mai mare decît contorul specificat. Valoarea sa este adăugată la indicatorul MARKER.

Pentru fișierele ce sînt editate, numărul actual de octeți nu poate fi mai mare decît lungimea bufferului de editare.

Numărul de octeți transferați este egal cu :

min (CONTOR,(LENGTH-MARKER))

Sfîrșitul de fișier este indicat cînd numărul de octeți este zero sau cînd numărul de octeți este mai mic decît contorul (exceptînd fișierele editate).

Apelarea rutinei READ se face astfel :

a) În limbaj de asamblare

```
         EXTRN   SFDX        ; LEGĂTURA CU PUNCTUL DE IN-
                             ; TRARE ÎN SFDX
READ     EQU     3           ; IDENTIFICATORUL RUTINEI READ
         MVI     C,READ      ; ÎN C NUMĂRUL DE IDENTIFICARE
                             ; RUTINĂ
         LXI     D,RBLK      ; ÎN D,E ADRESA BLOCULUI DE PA-
                             ; RAMETRI
         CALL    SFDX        ; APELARE RUTINĂ
         LDA     RSTAT       ; TEST INDICATOR EROARE
         ORA     A
         JNZ     ERR         ; SALT LA RUTINA DE TRATARE
         .                   ; A ERORII, DACĂ ESTE CAZUL
         .
         .
RBLK:                        ; BLOCUL DE PARAMETRI PENTRU
                             ; READ
RAFTN:   DS      2           ; NUMĂRUL LOGIC AL FIŞIERULUI
         DW      RBUF        ; ADRESA ZONEI ÎN CARE SE CITEŞTE
RCNT:    DW      128         ; CONTOR DE OCTEŢI DE TRANSFERAT
         DW      ACTUAL      ; ADRESA CONTOR DE OCTETI TRANS-
                             ; FERATI
         DW      RSTAT       ; ADRESA INDICATOR STARE
         .
         .
         .
ACTUAL:  DS      2           ; CONTOR DE OCTEŢI TRANSFERAŢI
RSTAT:   DS      2           ; CONŢINE STAREA
RBUF:    DS      128         ; ZONA DE MEMORIE ÎN CARE SE
                             ; CITEŞTE
```

b) În limbajul PL/M

```
READ :
   PROCEDURE (AFTN, BUFFER, COUNT, ACTUAL, STATUS), EXTERNAL ;
     DECLARE (AFTN, BUFFER, COUNT, ACTUAL, STATUS) ADDRESS ;
END READ ;
   .
   .
   .
   DECLARE AFTN$IN ADDRESS ;
   DECLARE BUFFER (128) BYTE ;
   DECLARE ACTUAL ADDRESS ;
   DECLARE STATUS ADDRESS ;
   .
   .
   .
   CALL READ (AFTN$IN,.BUFFER, 128,.ACTUAL,.STATUS) ;
   .
   .
   .
```

6.3.5. WRITE — Transferă date din memorie într-un fişier

Rutina **WRITE** transferă date dintr-o zonă de memorie specificată într-un fişier deschis în prealabil cu OPEN. Blocul de control al funcţiei conţine 4 parametri, fig. 6.12 care au următoarea semnificaţie :

AFTN — conţine numărul logic al fişierului deschis pentru scriere, număr ce a fost fixat la deschidere de rutina OPEN.

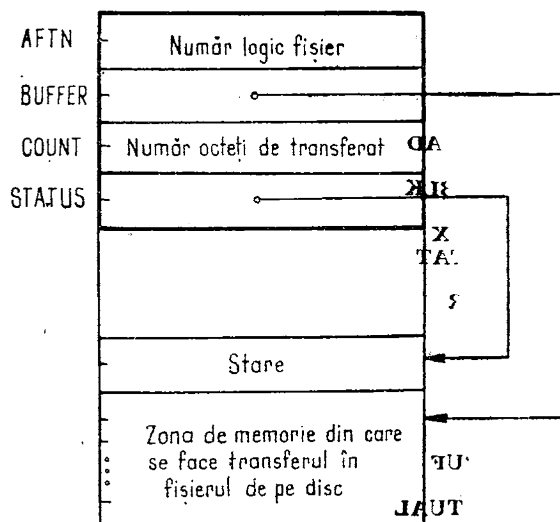


Fig. 6.12. Structura blocului de control pentru **funcţia** WRITE.

BUFFER — conţine adresa zonei de memorie din care datele vor fi transferate în fişier.

COUNT — conţine numărul de octeţi ce se vor transfera din zona de memorie în fişierul de ieşire.

STATUS — conţine adresa unei celule de memorie unde se depune eventualul cod de eroare rezultat în urma apelului rutinei. Dacă totul decurge normal la această adresă se va găsi 0.

De subliniat faptul că valoarea contorului este adăugată la indicatorul MARKER. Indicatorul LENGTH este actualizat în mod corespunzător, numărul de octeţi transferaţi fiind egal cu contorul.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX        ; LEGATURA CU PUNCTUL DE IN-
                            ; TRARE IN SFDX
        EQU     4           ; IDENTIFICATORUL RUTINEI WRITE
        MVI     C,WRITE     ; ÎN C NUMARUL DE IDENTIFICARE
                            ; RUTINA
        LXI     D,WBLK      ; ÎN D,E ADRESA BLOCULUI DE PA-
                            ; RAMETRI
        CALL    SFDX        ; APELARE RUTINA
        LDA     WSTAT       ; TEST INDICATOR EROARE
        ORA     A
        JNZ     ERR         ; SALT LA RUTINA DE TRATARE A
        .                   ; ERORII, DACA ESTE CAZUL
        .
        .
                            ; BLOCUL DE PARAMETRI PENTRU
WBLK:                       ; WRITE
                            ; NUMARUL LOGIC AL FISIERULUI
        DS      2           ; ADRESA ZONEI DIN CARE SE SCRIE
        DW      WBUF        ; IN FISIER
```

```
WCNT:   DW      128             ; CONTORUL DE OCTETI DE TRANS-
                                ; FERAT
        DW      WSTAT           ; ADRESA INDICATOR STARE
        .
        .
        .
WSTAT:  DS      2               ; CONTINE STAREA
WBUF:   DS      128             ; ZONA DE MEMORIE DIN CARE ARE
                                ; LOC TRANSFERUL IN FISIER
```

b) În limbajul PL/M

```
WRITE:
    PROCEDURE (AFTN, BUFFER, COUNT, STATUS) EXTERNAL;
      DECLARE (AFTN, BUFFER, COUNT, STATUS) ADDRESS;
    END WRITE;
    .
    .
    .
    DECLARE AFT$IN ADDRESS;
    DECLARE BUFFER (128) BYTE;
    DECLARE STATUS ADDRESS;
    .
    .
    .
    CALL WRITE(AFT$IN,.BUFFER,128,.STATUS)
    .
    .
    .
```

6.3.6. SEEK — Poziționează indicatorul MARKER asociat unui fișier

Rutina **SEEK** permite aflarea valorii sau schimbarea valorii indicatorului MARKER asociat unui fișier deschis anterior pentru intrare sau actualizare.

Indicatorul MARKER poate fi modificat în patru moduri :

— înainte
— înapoi
— la o locație specificată
— la sfîrșitul fișierului

Blocul de control al funcției conține 5 parametri, fig. 6.13, care au următoarea semnificație :

AFTN — conține numărul logic al fișierului deschis pentru citire sau actualizare, număr ce a fost fixat de rutina OPEN.

MODE — specifică acțiunea ce va fi aplicată asupra indicatorului MARKER. Astfel :

MODE = 0 specifică faptul că valoarea curentă a indicatorului MARKER reprezentată de numărul blocului și numărul octetului va fi returnată în urma apelării rutinei SEEK. Numărul blocului va fi depus la adresa indicată de parametrul BLOCKNO iar numărul octetului va fi depus la adresa indicată de parametrul BYTENO.

MODE = 1 specifică mutarea înapoi a indicatorului MARKER (spre începutul fișierului) cu un număr de octeți calculat pe baza parametrilor BLOCKNO și BYTENO.

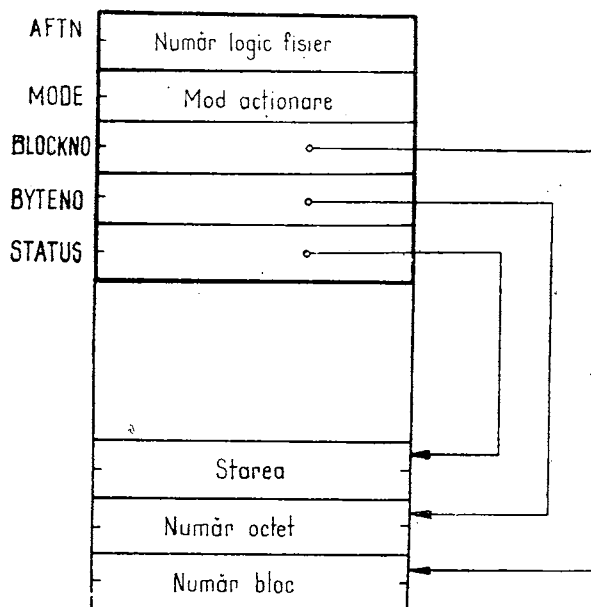


Fig. 6.13. Structura blocului de control pentru funcţia SEEK.

MODE = 2 specifică mutarea absolută a indicatorului MARKER la valoarea calculată pe baza parametrilor BLOCKNO şi BYTENO.

MODE = 3 specifică mutarea înainte a indicatorului MARKER (spre sfîrşitul fişierului) cu un număr de octeţi calculat pe baza parametrilor BLOCKNO şi BYTENO.

MODE = 4 specifică mutarea indicatorului MARKER la sfîrşitul fişierului (MARKER va fi egal cu LENGTH).

BLOCKNO — conţine adresa unei zone de memorie de doi octeţi utilizată pentru păstrarea numărului blocului, în operaţii de modificare a indicatorului MARKER.

BYTENO — conţine adresa unei zone de memorie de doi octeţi utilizată pentru păstrarea numărului octeţilor în operaţii de modificare a indicatorului MARKER.

STATUS — conţine adresa unei zone de memorie de doi octeţi în care SFDX va memora numărul eventualei erori rezultată în urma apelării rutinei SEEK. Dacă totul decurge normal la această adresă se găseşte 0.

Numărul de octeţi (N) cu care se modifică relativ sau absolut indicatorul MARKER se calculează astfel :

N = 128 * număr bloc + număr octet.

În cazul mutării înapoi a indicatorului MARKER dacă numărul de octeţi calculat (N) tinde să mute MARKER-ul înainte de începutul fişierului, valoarea indicatorului se poziţionează pe 0 şi se indică o eroare nefatală.

În cazul mutării absolute a indicatorului MARKER (MODE = 2) dacă numărul de octeţi, calculat (N) este 0 are loc o poziţionare a indicatorului la început de fişier. În cazul în care mutarea indicatorului are loc peste valoarea LENGTH, fişierul va fi completat cu zerouri (0H) astfel încît MARKER să fie egal cu LENGTH. Acelaşi lucru se întîmplă şi în cazul mutării înainte cînd are loc depăşirea sfîrşitului de fişier.

Apelarea rutinei SEEK se poate face astfel :

a) În limbaj de asamblare

```
          EXTRN     SFDX          ; LEGATURA CU PUNCTUL DE IN-
                                  ; TRARE IN SFDX
SEEK      EQU       5             ; IDENTIFICATORUL RUTINEI SEEK
          MVI       C,SEEK        ; ÎN C NUMARUL DE IDENTIFICARE
                                  ; RUTINA
          LXI       D,SBLK        ; ÎN D,E ADRESA BLOCULUI DE PA-
                                  ; RAMETRI
          CALL      SFDX          ; APELARE RUTINA.
          LDA       SSTAT         ; TEST INDICATOR EROARE
          ORA       A
          JNZ       ERR           ; SALT LA RUTINA DE TRATARE A
          .                       ; ERORII, DACA ESTE CAZUL
          .
          .
SBLK:                             ; BLOCUL DE PARAMETRI AI RUTINEI
                                  ; SEEK
SAFT:     DS        2             ; NUMARUL LOGIC AL FISIERULUI
MODE:     DS        2             ; MODUL DE ACTIONARE ASUPRA
                                  ; MARKER
BLOCKNO:  DW        BLKS          ; ADRESA PARAMETRULUI NUMAR
                                  ; BLOC
BYTENO:   DW        NBYTE         ; ADRESA PARAMETRULUI NUMĂR
                                  ; OCTET
STATUS:   DW        SSTAT         ; ADRESA INDICATOR EROARE
BKLS:     DS        2             ; NUMARUL BLOC
NBYTE:    DS        2             ; NUMARUL OCTET
SSTAT:    DS        2             ; INDICATOR STARE
```

b) În limbajul PL/M

```
SEEK :
    PROCEDURE (AFTN, MODE, BLOCKNO, BYTENO, STATUS) EXTERNAL ;
      DECLARE (AFTN, MOTE, BLOCKNO, BYTENO, STATUS) ADDRESS ;
    END SEEK ;
    .
    .
    .
    DECLARE AFT$IN ADDRESS ;
    DECLARE BLOCKNO ADDRESS ;
    DECLARE BYTENO ADDRESS ;
    DECLARE STATUS ADDRESS ;
    .
    .
    .
    CALL SEEK (AFT$IN, 0,.BLOCKNO,.BYTENO,.STATUS) ;
    .
    .
    .
```

6.3.7. LOAD — Încarcă și lansează în execuție un program

Rutina **LOAD** încarcă un fișier obiect absolut și transferă controlul fie programului încărcat, fie monitorului, fie programului apelant, în funcție de un parametru.

Blocul de control al funcției conține 5 parametri, fig. 6.14, care au următoarea semnificație :

FILE — conține adresa către o zonă de memorie ce conține numele fișierului sub formă de caractere ASCII. Numele fișierului trebuie să fie urmat de un caracter special diferit de două puncte (:) sau punct (.). De obicei se termină cu spațiu (b̶) sau (CR).

BIAS — specifică o valoare de deplasament ce se adună la adresa de încărcare a programului obiect absolut. Uzual un program încărcat cu un deplasament oarecare nu poate fi executat la această adresă.

SWITCH — conține o valoare ce specifică unde se transferă controlul după încărcare

SWITCH = 0 transferă controlul programului ce a apelat rutina LOAD.

SWITCH = 1 transferă controlul programului încărcat la adresa specificată în ultima înregistrare

SWITCH = 2 transferă controlul monitorului

ENTRY — conține adresa unei zone de memorie de doi octeți în care se depune valoarea de start (lansare) a programului încărcat cînd parametrul de control SWITCH=0. Dacă programul încărcat nu este un program principal se depune valoarea 0.

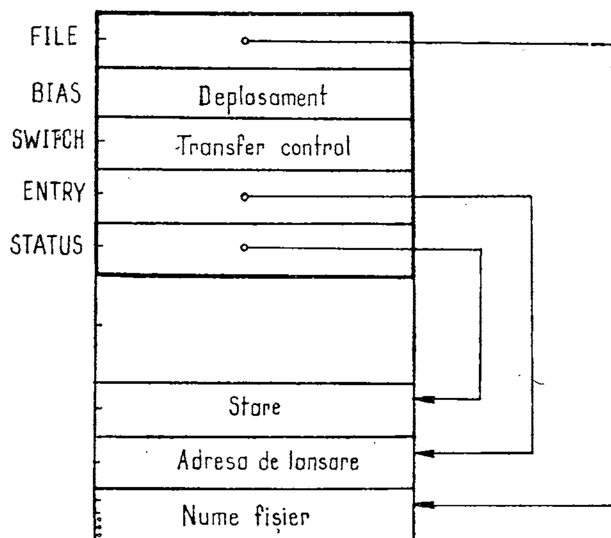


Fig. 6.14. Structura blocului de control pentru funcția LOAD.

STATUS — conține adresa unei celule de memorie unde la apelarea rutinei LOAD se depune numărul erorii nefatale, ce a apărut în timpul execuției sale, sau 0 dacă totul a decurs normal.

Apelarea rutinei LOAD se face astfel :

```
        EXTRN   SFDX         ; LEGATURA CU PUNCTUL DE IN-
                             ; TRARE IN SFDX
LOAD    EQU     6            ; IDENTIFICATORUL RUTINEI LOAD
        MVI     C,LOAD       ; ÎN C NUMARUL DE IDENTIFICARE
                             ; RUTINA
        LXI     D,LBLK       ; ÎN D,E ADRESA BLOCULUI DE PA-
                             ; RAMETRI
        CALL    SFDX         ; APELAREA RUTINEI
        LDA     LSTAT        ; TEST INDICATOR EROARE
        ORA     A            ; CONTINE DEPLASAMENTUL
        JNZ     ERR          ; SALT LA RUTINA DE TRATARE
.
.
.
LBLK:                        ; BLOCUL DE PARAMETRI PENTRU
                             ; LOAD
        DW      FILNAM       ; ADRESA ZONEI NUMELUI DE FISIER
BIAS:   DS      2
SWIRCH: DS      2            ; CONTINE PARAMETRUL DE TRANS-
                             ; FER CONTROL
        DW      ENAD         ; ADRESA INDICATOR EROARE
        DW      LSTAT        ; ADRESA PUNCTULUI DE LANSARE
FILNAM: DS      15           ; CONTINE NUMELE FISIERULUI
ENAD:   DS      2            ; CONTINE ADRESA DE LANSARE
LSTAT:  DS      2            ; STAREA
```

b) În limbajul PL/M

```
LOAD:
    PROCEDURE (FILE, BIAS, SWITCH, ENTRY, STATUS) EXTERNAL;
      DECLARE (FILE, BIAS, SWITCH, ENTRY, STATUS) ADDRESS;
    END LOAD;
    .
    .
    .
    DECLARE FILNAM (15) BYTE;
    DECLARE ENTRY ADDRESS;
    DECLARE STAT$IN ADDRESS;
    .
    .
    .
    CALL LOAD (.FILNAM,0,1,.ENTRY,.STAT$IN);
```

6.3.8. RENAME — Schimbă numele unui fișier de pe discul flexibil

Rutina **RENAME** schimbă numele unui fișier de pe discul flexibil cu un alt nume specificat de utilizator.

Blocul de control al funcției conține 3 parametri, fig. 6.15, care au următoarea semnificație :

OLDFILE — specifică adresa de început a zonei de memorie care conține numele vechi al fișierului sub formă de caractere ASCII. Numele fișierului trebuie urmat de un caracter special diferite de două puncte (:) sau punct (.). De obicei se termină cu spațu (␢) sau (CR).

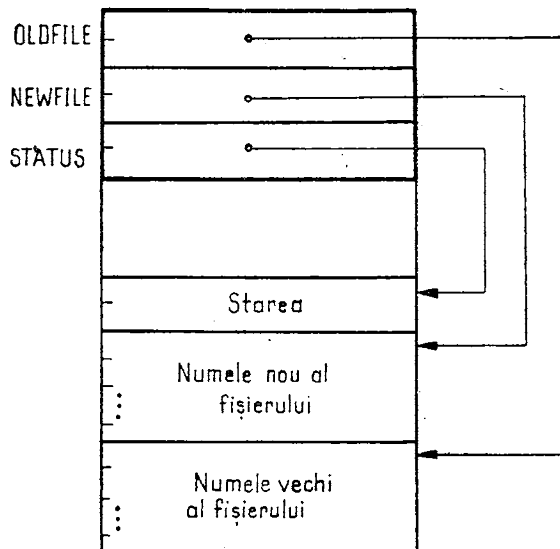


Fig. 6.15. Structura blocului de control pentru funcția RENAME.

NEWFILE — specifică adresa de început a zonei de memorie care conține numele nou al fișierului sub formă de caractere ASCII. Numele fișierului trebuie urmat de un caracter special diferit de două puncte (:) sau punct (.). De obicei se termină cu spațiu (␢) sau (CR).

STATUS — conține adresa unei celule de memorie unde se depune numărul erorii nefatale ce eventual poate să apară în timpul execuției acestei rutine.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
         EXTRN   SFDX        ; LEGATURA CU PUNCTUL DE IN-
                             ; TRARE IN SFDX
RENAME   EQU     7           ; IDENTIFICATORUL RUTINEI RENAME
         MVI     C,RENAME    ; ÎN C NUMARUL DE IDENTIFICARE
                             ; RUTINA
         LXI     D,RNBLK     ; ÎN D,E ADRESA BLOCULUI DE PA-
                             ; RAMETRI
         CALL    SFDX        ; APELARE RUTINA
         LDA     RNSTAT      ; TEST INDICATOR EROARE
         ORA     A
         JNZ     ERR         ; SALT LA RUTINA DE TRATARE A
                             ; ERORII, DACA ESTE CAZUL
.
.
.
RNBLK:                       ; BLOCUL DE PARAMETRI PENTRU
                             ; RENAME
         DW      FILE2       ; ADRESA SPRE NUMELE VECHI AL
                             ; FISIERULUI
         DW      FILE1       ; ADRESA SPRE NUMELE NOU AL FI-
                             ; SIERULUI
```

```
         DW      RNSTAT          ; ADRESA INDICATOR EROARE
RNSTAT:  DS      2               ; CONŢINE STAREA (EVENTUAL NR.
                                 ; ERORII)
FILE1:   DB      'FILE.NEW '     ; NUMELE VECHI AL FIŞIERULUI
FILE2:   DB      'FILE.OLD '     ; NUMELE VECHI AL FIŞIERULUI
```

b) În limbajul PL/M

```
RENAME :
    PROCEDURE (OLDFILE, NEWFILE, STATUS) EXTERNAL ;
      DECLARE (OLDFILE, NEWFILE, STATUS) ADRESS ;
    END RENAME ;
    .
    .
    .
    DECLARE OFILE (20) BYTE ;
    DECLARE NFILE (20) BYTE ;
    DECLARE STAT$IN ADDRESS ;
    .
    .
    .
    CALL RENAME (.OFILE,.NFILE,.STAT$IN)
    .
    .
    .
```

6.3.9. CONSOL — Schimbă consola sistemului

Rutina **CONSOL** permite schimbarea echipamentului periferic utilizat ca şi consolă.

Blocul de control al funcţiei conţine 3 parametri, fig. 6.16, care au următoarea semnificaţie :

INFILE — specifică adresa de început a unei zone de memorie care conţine numele fişierului, sub formă de caractere ASCII, ce va fi utilizat ca intrare consolă.

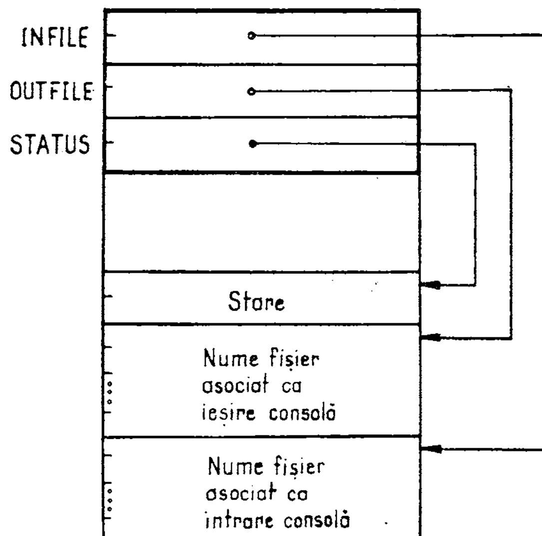


Fig. 6.16. Structura blocului de control pentru funcţia CONSOL.

OUTFILE — specifică adresa de început a unei zone de memorie care conține numele fișierului, sub formă de caractere ASCII, ce va fi utilizat ca ieșire consolă.

STATUS — conține adresa unei celule de memorie unde se va depune eventualul cod de eroare se rezultă în urma apelării rutinei.

Înaintea deschiderii noului fișier de intrare sau de ieșire se închide fișierul curent. Excepție fac fișierele :CI: respectiv :CO: care rămîn tot timpul deschise.

Apelarea rutinei CONSOL se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX         ; LEGATURA CU PUNCTUL DE IN-
                             ; TRARE IN SFDX
CONSOL  EQU     8            ; IDENTIFICATORUL RUTINEI CONSOL
        MVI     C,CONSOL     ; ÎN C NUMARUL DE IDENTIFICARE
                             ; RUTINA
        LXI     D,CBLK       ; ÎN D,E ADRESA BLOCULUI DE PA-
                             ; RAMETRI
        CALL    SFDX         ; APELARE RUTINA
        LDA     CSTAT        ; TEST INDICATOR EROARE
        ORA     A
        JNZ     ERR          ; SALT LA RUTINA DE TRATARE A
.                            ; ERORII, DACA ESTE CAZUL
.
.
CBLK:                        ; BLOCUL DE PARAMETRI PENTRU
                             ; CONSOL
        DW      INFILE       ; ADRESA NUME FISIER INTRARE
                             ; CONSOLA
        DW      OTFILE       ; ADRESA NUME FISIER IESIRE CON-
                             ; SOLA
        DW      CSTAT        ; ADRESA INDICATOR EROARE
.
.
.
INFILE: DS      15           ; CONTINE NUME FISIER INTRARE
                             ; CONSOLA
OTFILE: DS      15           ; CONTINE NUME FISIER IESIRE
                             ; CONSOLA
CSTAT:  DS      2            ; CONTINE STAREA
```

b) În limbajul PL/M

```
CONSOL:
   PROCEDURE (INFILE,OUTFILE,STATUS)EXTERN ;
   DECLARE (INFILE, OUTFILE,STATUS)ADDRESS ;
   END CONSOLE ;
.
.
.
DECLARE INFILE(15)BYTE ;
DECLARE OUTFILE(15)BYTE ;
DACLARE STAT$IN ADDRESS ;
.
.
.
CALL CONSOL(.INFILE,.OUTFILE,.STAT$IN) ;
.
.
.
```

6.3.10. EXIT — Transferă controlul din programul utilizator sistemului de operare

Rutina **EXIT** apelată din programele utilizator închide toate fişierele deschise, cu excepţia fişierelor :CI: şi :CO: care sînt tot timpul deschise, şi transferă controlul sistemului de operare în regim de primire comenzi.

Consola sistemului asignată în mod curent nu este modificată.

Blocul de control al funcţiei conţine un parametru, fig. 6.17 care are următoarea semnificaţie :

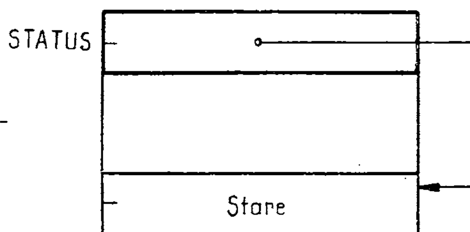


Fig. 617. Structura blocului de control pentru funcţia EXIT.

STATUS — conţine adresa unei celule de memorie unde se depune eventuala eroare ce apare în urma apelării rutinei. (În general nu are nici o semnificaţie).

Apelarea rutinei se poate face astfel :

a) În limbaj de asamblare

```
         EXTRN   SFDX        ; LEGATURA CU PUNCTUL DE IN-
                             ; TRARE IN SFDX
EXIT     EQU     9           ; IDENTIFICATORUL RUTINEI EXIT
         MVI     C,EXIT      ; ÎN C NUMĂRUL DE IDENTIFICARE
                             ; RUTINA
         LXI     D,EBLK      ; ÎN D,E ADRESA BLOCULUI DE PA-
                             ; RAMETRI
         CALL    SFDX        ; APELARE RUTINĂ
EBLK :                       ; BLOCUL DE PARAMETRI PENTRU
                             ; EXIT
         DW      ESTAT       ; ADRESA INDICATOR STARE (EROARE)
ESTAT :  DS      2           ; CONŢINE STAREA
```

b) În limbajul PL/M

```
EXIT :
    PROCEDURE EXTERNAL ;
    END EXIT ;


CALL EXIT ;
.
.
.
```

6.3.11. ATTRIB — Schimbă atributele unui fişier de pe disc

Rutina **ATTRIB** este folosită de programele utilizator pentru a schimba atributele unui fişier de pe disc.

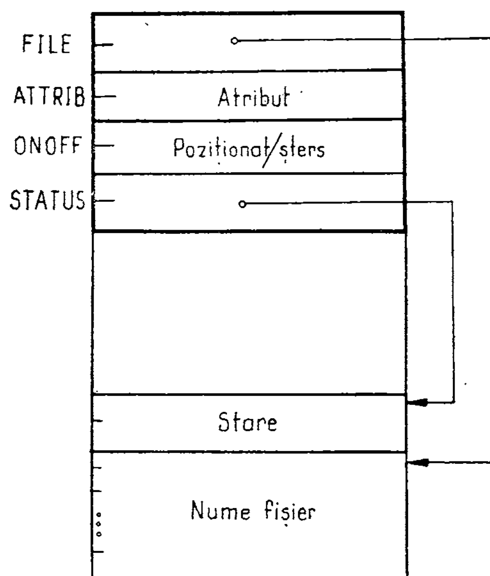


Fig. 6.18. Structura blocului de control pentru funcția ATTRIB.

Blocul de control al funcției conține 4 parametri, fig. 6.18, care au următoarea semnificație :

FILE — specifică adresa unei zone de memorie care conține numele fișierului, sub formă de caractere ASCII, ale cărui atribute se modifică prin apelarea rutinei.

Numele fișierului trebuie urmat de un caracter special diferit de două puncte (:) sau de punct (.). De obicei se termină cu spațiu (ƀ) sau (CR).

ATRIB — conține un identificator al atributului ce va fi schimbat.

Identificatorul poate avea valoarea :

0=specifică atributul invizibil (I)
1=specifică atributul sistem (S)
2=specifică atributul protejat la scriere (W)
3=specifică atributul format (F)

ONOFF — conține o valoare ce indică dacă atributul specificat de ATRIB va fi poziționat (va fi activat) sau va fi șters (dezactivat). Valoarea se memorează în octetul cel mai puțin semnificativ și are semnificația :

=0 arată că atributul va fi șters (dezactivat)
=1 arată că atributul va fi poziționat (activat)

STATUS — conține adresa unei locații de memorie unde se depune eventualul cod de eroare ce apare în urma apelării rutinei.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
         EXTRN    SFDX          ; LEGĂTURA CU PUNCTUL DE IN-
                                ; TRARE IN SFDX
ATTRIB   EQU      10            ; IDENTIFICATORUL RUTINEI ATTRIB
         MVI      C,ATTRIB      ; ÎN C NUMĂRUL DE IDENTIFICARE
                                ; RUTINA
```

```
         LXI      D,ABLK        RAMETRI
                                ; RAMETRI
         CALL     SFDX          ; APELARE RUTINĂ
         LDA      ASTAT         ; TEST INDICATOR EROARE
         ORA      A
         JNZ      ERR           ; SALT LA RUTINA DE TRATARE A
                                ; ERORII DACA ESTE CAZUL
ABLK:                           ; BLOCUL DE PARAMETRI PENTRU
                                ; ATTRIB
         DW       FILEN         ; ADRESA NUME FIŞIER
         DW       2             ; IDENTIFICATOR ATRIBUT (ÎN ACEST
                                ; EXEMPLU W)
         DW       0             ; IDENTIFICATOR VALOARE (DEZAC-
                                ; TIVARE, STERS)
         DW       ASTAT         ; ADRESA INDICATOR EROARE
FILEN:   DS       20            ; CONŢINE NUME FIŞIER
ASTAT:   DS       2             ; CONŢINE STAREA RETURNATĂ
```

b) În limbajul PL/M

```
ATTRIB:
   PROCEDURE(FILE,ATRIB,ONOFF,STATUS)EXTERNAL ;
     DECLARE(FILE,ATTRIB,ONOFF,STATUS)ADDRESS ;
   END ATTRIB ;
.
:
.
DECLARE FILE(20)BYTE ;
DECLARE STAT$IN ADDRESS ;
.
:
:
CALL ATTRIB(.FILE,2,0,.STAT$IN)
.
:
:
```

6.3.12. RESCAN — Poziţionează indicatorul MARKER la începutul liniei

Rutina **RESCAN** este utilizată numai cu fişierele de editat linie cu linie. Apelarea rutinei RESCAN va muta indicatorul MARKER la înce-

Fig. 6.19. Structura blocului de control pentru funcţia RESCAN.

putul liniei (logice) citite. O eventuală apelare a rutinei READ nu va produce o citire din fişierul de intrare ci va citi din zona tampon de editare din memorie.

Blocul de control al funcţiei conţine 2 parametri, fig. 6.19 care au următoarea semnificaţie :

AFTN — conţine numărul logic al fişierului deschis pentru editare linie cu linie, număr ce a fost fixat la deschidere de rutina OPEN.

STATUS — conţine adresa unei locaţii de memorie unde se va depune eventualul cod de eroare.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
         SFDX     EXTRN        ; LEGĂTURA CU PUNCTUL DE IN-
                               ; TRARE IN SFDX
RESCAN   EQU      11           ; IDENTIFICATORUL RUTINEI RES-
                               ; CAN
         MVI      C,RESCAN     ; ÎN C NUMĂRUL DE IDENTIFICARE
                               ; RUTINA
         LXI      D,REBLK      ; ÎN D,E ADRESA BLOCULUI DE PA-
                               ; RAMETRI
         LDA      RESTAT       ; TEST INDICATOR EROARE
         ORA      A
         JNZ      ERR          ; SALT LA RUTINA DE TRATARE A
                               ; ERORII DACA ESTE CAZUL
           :
REBLK:                         ; BLOCUL DE PARAMETRI PENTRU
                               ; RESCAN
         DS       2            ; NUMĂR LOGIC AL FIŞIERULUI
         DW       RESTAT       ; ADRESA INDICATOR EROARE
RESTAT:  DS       2            ; CONŢINE STAREA (EROARE)
```

b) În limbajul PL/M

```
RESCAN:
    PROCEDURE(AFTN,STATUS)EXTERNAL ;
      DECLARE(AFTN,STATUS)ADDRESS ;
      END RESCAN
.
.
.
DECLARE AFTN$IN ADDRESS ;
DECLARE STATUS ADDRESS ;
.
.
.
CALL RESCAN(AFTN$IN,.STATUS)
.
.
.
```

6.3.13. ERROR — Transmite mesaj de eroare la consola sistemului

Rutina **ERROR** scrie un mesaj de eroare (numărul erorii e specificat de utilizator) la consola iniţială a sistemului.

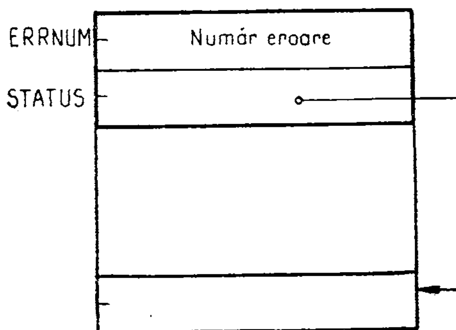


Fig. 6.20. Structura blocului de control pentru funcția ERROR

Blocul de control al funcției conține 2 parametri, fig. 6.20, care au următoarea semnificație :

ERRNUM — conține numărul erorii ce se va tipări la consola inițială a sistemului.

STATUS — adresa unei celule de memorie unde se va depune starea de desfășurare a apelului (în general nu se analizează de utilizator).

Numărul erorii specificat de parametrul ERRNUM trebuie să fie în cel mai puțin semnificativ octet al parametrului.

Numerele 0÷100 și 200÷255 sînt rezervate pentru programe de sistem iar numerele între 100÷199 pot fi folosite pentru programul utilizator.

Sistemul afișează mesajul de eroare sub forma

ERROR nnn USER PC mmmm

unde :

nnn — este numărul erorii specificate de utilizator în cadrul parametrului ERRNUM

mmmm — este adresa de întoarcere în programul care apelează rutina.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX        ; LEGĂTURA CU PUNCTUL DE IN-
                            ; TRARE IN SFDX
ERROR   EQU     12          ; IDENTIFICATORUL RUTINEI ERROR
        MVI     C,ERROR     ; ÎN C NUMĂRUL DE IDENTIFICARE
                            ; RUTINA
        LXI     D,EBLK      ; ÎN D,E ADRESA BLOCULUI DE PA-
                            ; RAMETRI
        CALL    SFDX        ; APELARE RUTINĂ
          :
EBLK:                       ; BLOCUL DE PARAMETRI PENTRU
                            ; RUTINA ERROR
        DS      2           ; CONȚINE NUMĂRUL ERORII SPECI-
                            ; FICATE
        DW      ESTAT       ; ADRESA INDICATOR STARE
ESTAT:  DS      2           ; CONȚINE STAREA RETURNATĂ
```

b) În limbajul PL/M

```
ERROR :
    PROCEDURE(ERRNUM)EXTERNAL ;
      DECLARE(ERRNUM)ADDRESS ;
    END ERROR ;
.

DECLARE ENUM(2)BYTE ;


CALL ERROR(ENUM) ;
```

6.3.14. WHOCON — Determină fişierul asignat drept consolă sistem

Rutina **WHOCON** determină care fişier este asignat drept consolă sistem (de intrare/ieşire).

Blocul de control al funcţiei conţine 3 parametri, fig. 6.21, care au următoarea semnificaţie :

AFTN — conţine adresa către numărul logic al fişierului asociat consolei, număr logic ce poate lua valorile :
0=specifică :CO: (ieşire consolă)
1=specifică :CI: (intrare consolă)
Parametrul indică pentru cine va returna numele fişierului, pentru intrare consolă (:CI:) sau ieşire consolă (:CO:).

BUFFER — specifică adresa unei zone de memorie, de 15 octeţi unde rutina WHOCON va returna numele fişierului asignat ca :CI: sau :CO:

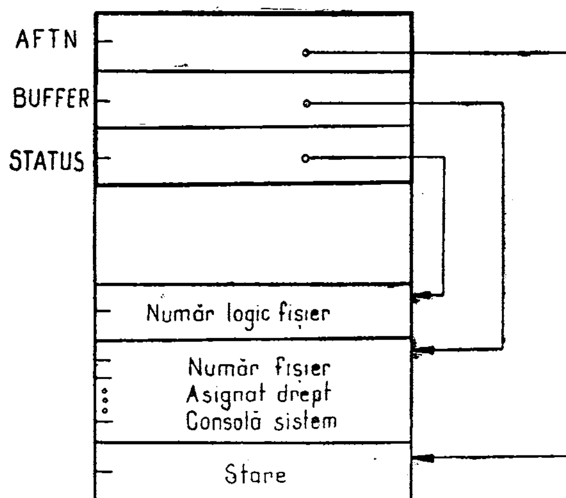


Fig. 6.21. Structura blocului de control pentru funcţia WHOCON.

Numele e returnat ca un şir de caractere ASCII terminat cu un spaţiu.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX        ; LEGĂTURA CU PUNCTUL DE IN-
                            ; TRARE IN SFDX
WHOCON  EQU     13          ; IDENTIFICATORUL RUTINEI WHO-
                            ; CON
        MVI     C,WHOCON    ; ÎN C NUMĂRUL DE IDENTIFICARE
                            ; RUTINA
        LXI     D,WHBLK     ; ÎN D,E ADRESA BLOCULUI DE PA-
                            ; RAMETRI
        CALL    SFDX        ; APEL RUTINA
.
.
.
WHBLK:                      ; BLOCUL DE PARAMETRI PENTRU
                            ; WHOCON
        DW      AFTN        ; ADRESA NUMĂR LOGIC CONSOLA
        DW      BUFIN       ; ADRESA ZONA UNDE SE DEPUNE
                            ; NUME FISIER
        DW      WHSTAT      ; ADRESA INDICATOR EROARE
AFTN:   DS      2           ; NUMĂR LOGIC CONSOLĂ 0 PT:CO:,
                            ; 1.PT:CI:
BUFIN:  DS      15          ; SE DEPUNE NUME FIŞIER
WHSTAT: DS      2           ; CONŢINE STAREA
```

b) În limbajul PL/M

```
WHOCON:
   PROCEDURE(AFTN,BUFFER)EXTERNAL ;
     DECLARE(AFTN,BUFFER)ADDRESS ;
   END WHOCON ;
.
.
.
DECLARE BUFF$IN(15)BYTE ;
.
.
.
CALL WHOCON(1,.BUFF$IN) ;
.
.
.
```

6.3.15. SPATH — Stabileşte informaţii despre un fişier

Rutina **SPATH** obţine informaţii despre un fişier specificat de utilizator, referitoare la :

— numărul perifericului pe care este rezident fişierul ;
— numele fişierului şi extensia sa ;
— tipul perifericului ;
— tipul unităţii de disc.

Blocul de control al funcţiei conţine 3 parametri, fig. 6.22, care au următoarea semnificaţie :

FILE — specifică adresa unei zone de memorie care conţine numele fişierului pentru care se doresc informaţii. Numele este sub forma unui

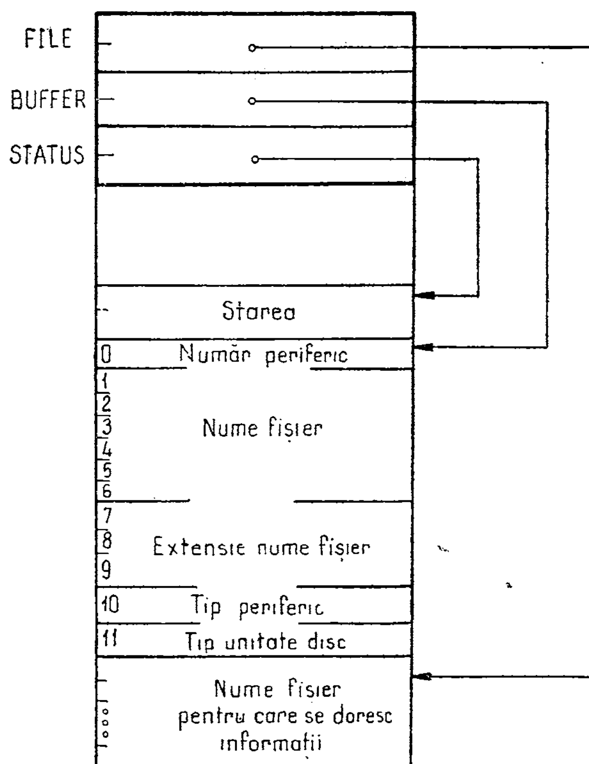


Fig. 6.22. Structura blocului de control pentru funcţia SPATH.

şir de caractere ASCII, ce se termină cu un caracter special diferit de două puncte (:) sau punct (.). De obicei numele se termină cu spaţiu (ƀ) sau (CR).

BUFFER — conţine adresa unei zone de memorie în care va fi returnată informaţia despre fişierul specificat de utilizator prin parametrul AFTN.

STATUS — conţine adresa unei locaţii de memorie în care se depune eventualul cod de eroare nefatală rezultat în urma apelului rutinei SPATH.

Informaţia din zona de memorie specificată de parametrul BUFFER are următoarea structură:

octet 0 — număr periferic — conţine un număr ce reprezintă identificatorul echipamentului periferic ce constituie suport pentru fişierul specificat. Poate lua una din valorile:

0 — unitatea de disc 0 (:F0:)
1 — unitatea de disc 1 (:F1:)
2 — unitatea de disc 2 (:F2:)
3 — unitatea de disc 3 (:F3:)
6 — intrare de la interfaţa serială TTY (:TI:)
7 — ieşire la interfaţa serială TTY (:TO:)

8 — intrare de la interfața serială USART (:VI:)
9 — ieșire la interfața serială USART (:VO:)
0AH — intrare de la consola nestandard, utilizator (:I1:)
0BH — ieșire la consola nestandard, utilizator (:01:)
0CH — cititor de bandă perforată (:PR:)
0DH — cititor de cartele (:CR:)
0EH — caseta magnetică (:R1:)
0FH — bandă magnetică (:R2:)
10H — ieșire la interfața serială TTY (:TO:)
11H — perforator de bandă (:PP:)
12H — casetă magnetică (:P1:)
13H — bandă magnetică (:P2:)
14H — imprimanta (:LP:)
15H — banda magnetică ca echipament list (:P2:)
16H — periferic de ieșire fictiv „byte bucket" (:BB:)
17H — intrare consolă curentă (:CI:)
18H — ieșire consolă curentă (:CO:)

octeți 1—9 — numele și extensia numelui fișierului, ca o succesiune de caractere ASCII.

octet 10 — tip periferic — specifică tipul perifericului ce constituie suport pentru fișierul specificat.

Poate lua valorile :
0 — periferic de ieșire secvențial ;
1 — periferic de intrare secvențial ;
2 — periferic de intrare/ieșire secvențial ;
3 — periferic de intrare/ieșire aleator.

octet 11 — tip unitate disc — are semnificație numai cînd tipul perifericului are valoarea 3 (periferic de intrare/ieșire aleator). Poate lua una din valorile :
0 — unitate de disc inexistentă ;
2 — unitate de disc simplă densitate.

Dacă tipul perifericului este diferit de 3, în acest octet se va găsi valoarea 0FFH.

Apelarea rutinei se face astfel :

a) În limbaj de asamblare

```
        EXTRN   SFDX        ; LEGĂTURA CU PUNCTUL DE IN-
                            ; TRARE IN SFDX
SPATH   EQU     14          ; IDENTIFICATORUL RUTINEI SPATH
        MVI     C,SPATH     ; ÎN C NUMĂR IDENTIFICARE RU-
                            ; TINA
        LXI     D,SPBLK     ; ÎN D,E ADRESA BLOCULUI DE PA-
                            ; RAMETRI
        CALL    SFDX        ; APELARE RUTINĂ
        LDA     SPSTAT      ; TESTARE INDICATOR EROARE
        ORA     A           ;
        JNZ     ERR         ; SALT LA RUTINA DE TRATARE A
        .                   ; ERORII, DACA ESTE CAZUL
        .
        .
SPBLK:                      ; BLOCUL DE PARAMETRI PENTRU
                            ; SPATH
        DW      FILEN       ; ADRESA NUME FIȘIER
```

```
         DW       BUFIN:     ; ADRESA INFORMAŢII DESPRE FI-
                             ; SIER
         DW       SPSTAT:    ; ADRESA INDICATOR EROARE
FILEN:   DS       15         ; CÎMP NUME FIŞIER
BUFIN:   DS       12         ; INFORMAŢII DESPRE FIŞIER
SPSTAT:  DS       2          ; INDICATOR STARE
```

b) În limbajul PL/M

```
SPATH :
PROCEDURE (FILE, BUFFER, STATUS) EXTERNAL ;
DECLARE (FILE, BUFFER, STATUS) ADDRESS ;
END SPATH ;
.
.
.
DECLARE FILENAM (15) BYTE ;
DECLARE BUF$IN (12) BYTE ;
DECLARE STAT$IN ADDRESS ;
.
.
.
CALL SPATH (.FILENAM, .BUF$IN, .STAT$IN) ;
```

6.4. Modelul operaţional pentru fazele de pregătire-execuţie a programelor sub SFDX-18

Sistemul de operare SFDX-18, permite o programare modulară ceea ce înlătură dificultăţile de scriere, punere la punct şi testare a programelor de complexitate ridicată. În acest sens sistemul permite pregătirea şi testarea individuală a modulelor ce alcătuiesc un program oferind utilizatorului programe de bază ca :

— editoare de texte ;

— translatoare care generează programe obiect în format relocatabil ;

— editor de legături care combină mai multe module într-un singur modul ;

— locator de adrese care atribuie adrese de memorie absolute unui program relocatabil ;

— bibliotecar care formează şi întreţine biblioteci de programe accesibile la editarea de legături ;

— încărcător de programe absolute.

Modelul operaţional pentru fazele de pregătire-execuţie a programelor sub SFDX-18 este prezentat în fig. 6.23.

Fluxul operaţiilor pentru fazele de pregătire-execuţie ar fi următorul :

— utilizatorul pregăteşte programele sursă (PS) pe un suport ca banda perforată, banda magnetică, caseta magnetică, discul magnetic utilizînd editoarele de texte ETX, EDIT, CREDIT ;

— se trece programul sursă pe discul magnetic utilizînd comenzi SFDX ;

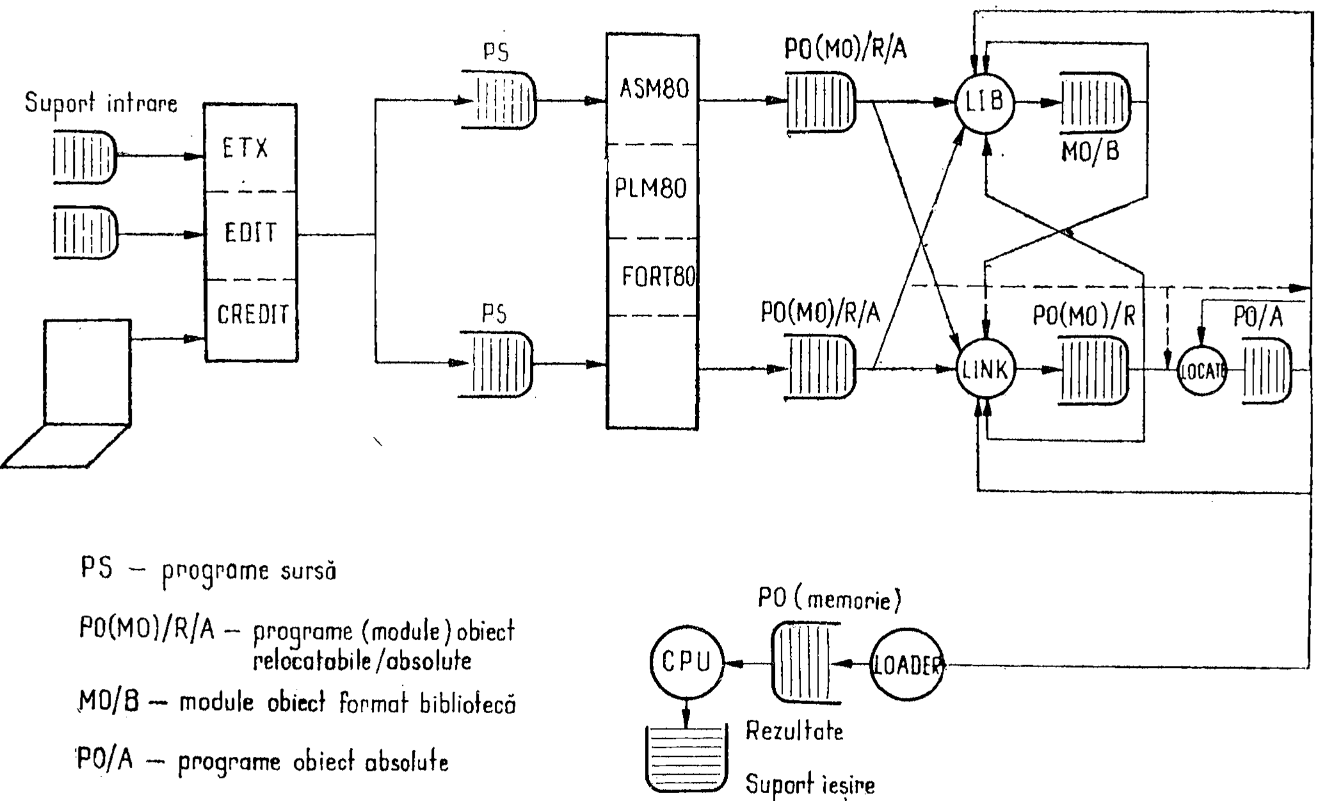


Fig. 6.23. Modelul operațional al fazelor de pregătire-execuție a programelor sub SFDX-18.

— asupra programelor sursă (PS) se acţionează cu translatoarele ASM80, PLM80, FORT80 etc. şi se obţin programe obiect (module obiect) relocatabile sau absolute (PO(MO)/RA) ;

— modulele obiect relocatabile (MO/R) pot fi introduse în biblioteca de module obiect cu ajutorul comenzii LIB, imediat după translatare sau după ce au fost legate cu alte module sau locatate ;

— modulele obiect obţinute în urma translatării pot fi legate între ele, cu module din bibliotecă sau cu module care au fost locatate cu ajutorul comenzii LINK, generîndu-se un singur modul ;

— programul (modulul) relocatabil poate primi adrese absolute sub acţiunea comenzii LOCATE obţinîndu-se un program (modul) absolut ce poate fi direct executabil ;

— programul obiect absolut (direct executabil) se încarcă în memorie cu comenzi SFDX, prin specificarea numelui său şi se lansează în execuţie de la adresa specificată în etapele de elaborare.

Cînd un program este mai mare decît spaţiul de memorie disponibil şi există module a căror execuţie este secvenţială, se pot înlănţui modulele fără să se combine într-un singur modul. Un program poate fi segmentat în secţiuni care se încarcă pentru execuţie în aceeaşi zonă de memorie.

Secţiunile de program care se încarcă separat trebuie să fie în fişiere separate. Încărcarea lor se face prin program cu apelul funcţiei de sistem LOAD sau cu rutine de I/E, utilizatorul trebuind să păstreze evidenţa segmentelor încărcate în memorie.

Pentru a realiza segmentarea programelor trebuie rezolvate două probleme şi anume :

— rezolvarea referinţelor externe ;

— stabilirea adreselor de memorie la care se încarcă fiecare segment.

Stabilirea adreselor de memorie la care se încarcă fiecare segment se realizează prin utilizarea repetată a comenzii LOCATE, iar rezolvarea referinţelor externe fără a combina modulele într-unul singur se realizează prin utilizarea comenzii LINK cu parametrul PUBLICS.

Exemplu :

Fie un modul rădăcină SR care apelează segmentele S1 şi S2, pe care le încarcă pentru execuţie în aceeaşi zonă de memorie însă la momente diferite de timp. Segmentul S1, în timpul execuţiei sale, apelează segmentele S11 şi S12 pe care le încarcă pentru execuţie în aceeaşi zonă de memorie însă la momente diferite de timp (S11, S12 nu sînt în memorie în acelaşi timp). În figura 6.24 se prezintă modul de înlănţuire a segmentelor şi referinţele la simboli dintr-un segment în altul.

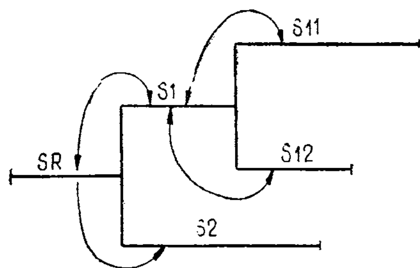


Fig. 6.24. Înlănţuirea segmentelor şi referinţele de simboli publici.

Presupunem că rădăcina are referințe externe la simbolii globali din S1 și S2. Segmentul S1 are referințe externe la simbolii publici din SR, S11 și S12. Segmentul S2 are referințe externe la simbolii publici din SR. Segmentele S11 și S12 au referințe la simbolii publici din S1.

Toate părțile structurii de suprapunere trebuie să fie în fișiere separate deoarece se încarcă separat.

Prin comanda :

— **LOCATE SR.REL TO SR.LOC MAP** ...

se determină adresa de bază pentru segmentele S1 și S2 și valorile absolute ale simbolilor publici din segmentul SR

Utilizînd această adresă de bază prin :

— **LOCATE S1.REL TO S1.LOC MAP CODE** (...) ...

— **LOCATE S2.REL TO S2.LOC CODE** (...) ...

se determină adresa de bază pentru segmentele S11 și S12 și valorile absolute ale simbolilor publici din segmentele S1 și S2.

Pentru a determina valorile absolute ale simbolilor publici din S11 și S12 se utilizează comenzile :

— **LOCATE S11.REL TO S11.LOC CODE** (...) ...

— **LOCATE S12.REL TO S12.LOC CODE** (...) ...

În urma acestor comenzi s-au obținut valorile absolute ale simbolilor publici din fiecare segment, fiecare segment are asignate adrese absolute de memorie dar referințele externe sînt nesatisfăcute.

Satisfacerea referințelor externe se realizează prin comenzi LINK cu parametrul PUBLICS.

Ținînd seama de referințele externe, prezentate în figura 6.24, secvența de comenzi LINK care satisface aceste referințe este :

— **LINK SR.LOC, PUBLICS (S1.LOC, S2.LOC) TO SR**

— **LINK S1.LOC, PUBLICS (SR.LOC, S11.LOC, S.12.LOC) TO S1**

— **LINK S2.LOC, PUBLICS (SR.LOC) TO S2**

— **LINK S11.LOC, PUBLICS (S1.LOC) TO S11**

— **LINK S12.LOC, PUBLICS (S1.LOC) TO S12**

Pentru a verifica dacă toate referințele externe sînt satisfăcute ar trebui să apelăm din nou funcția LOCATE asupra modulelor produse de LINK și să nu se mai obțină nici un mesaj de eroare privitor la referințe externe nesatisfăcute.

Trebuie subliniat faptul că programul utilizator trebuie să-și gestioneze singur suprapunerile. Înainte de a începe execuția unui segment trebuie ca acesta să fie încărcat în memorie. Dacă un segment conține zonă de date produsă de el, această zonă trebuie salvată pe disc înainte de a se încărca noul segment care se suprapune cu el.

Gestiunea suprapunerilor în timpul execuției unui program trebuie realizată de utilizator.

Capitolul 7

Editarea textelor

În acest capitol sînt prezentate posibilitățile de editare a textelor utilizînd editoarele ETX18 — pe microcalculatorul M18 fără disc, EDIT și CREDIT — pe M18, M18B și M118 prevăzute cu cel puțin o unitate de discuri flexibile.

7.1. Editorul de texte ETX18

7.1.1. Prezentarea generală a editorului

ETX18 reprezintă un program care permite utilizatorului să creeze și să actualizeze orice text sursă, format din șiruri de caractere ASCII. Ca mod de lucru, ETX18 este un editor prin context, orientat pe caractere și linii de text sursă. Pentru ca editorul să opereze corect, nu este nevoie să se adauge textului numerotarea liniilor sau alte informații ajutătoare. ETX18 permite crearea, direct de la consolă, a unui program sursă, corectarea lui prin ștergerea, adăugarea sau înlocuirea unui caracter sau șir de caractere dintr-o linie, ștergerea, adăugarea sau înlocuirea unor linii din text, salvarea pe bandă a textelor editate etc.

Toate funcțiile editorului sînt activate în mod interactiv, prin comenzi de la consola sistemului și au efect asupra zonei de lucru. Configurația minimă pentru execuția editorului trebuie să includă un echipament READER, un echipament PUNCH și cel puțin 8 Kocteți de memorie RAM. Din memoria RAM disponibilă, editorul ocupă aproximativ 4 Kocteți, restul fiind utilizată ca zonă de lucru — Zona Text. Dimensiunea Zonei Text este variabilă, crește pe măsură ce se introduce textul și scade pe măsură ce se șterge textul. Dacă Zona Text este vidă, sfîrșitul Zonei Text coincide cu începutul ei, avînd dimensiunea zero. Zona Text nu poate depăși zona de lucru disponibilă.

Fiind orientat pe caractere, este nevoie de un Indicator în Zona Text pentru a arăta caracterul care va fi afectat de o anumită comandă. Acest indicator nu apare în mod explicit scris la consolă și este poziționat întotdeauna fie între două caractere adiacente, fie înaintea primului ca-

racter din Zona Text, fie după ultimul caracter din Zona Text. Indicatorul nu va fi niciodată poziționat pe un caracter.

Considerînd o linie ca un șir de caractere care se termină cu Line Feed (0AH), Indicatorul poate fi deplasat, prin comenzi, peste caractere sau peste linii. Dacă nu există nici un caracter Line Feed întregul text este considerat o linie. De asemenea, la introducerea unui caracter dintr-un text sursă, Indicatorul se deplasează cu o poziție. Indicatorul nu poate fi deplasat în afara Zonei Text. O încercare de deplasare a Indicatorului peste limitele Zonei, va avea ca efect terminarea comenzii în momentul cînd Indicatorul ajunge la una din aceste limite.

7.1.2. Comenzile editorului

Pentru a face interacțiunea utilizatorului cu editorul cît mai eficientă, s-a ales o sintaxă simplă pentru comenzi. Mnemonicele comenzilor sînt formate dintr-o singură literă. Unele comenzi necesită parametri. Pentru a facilita lucrul cu editorul, utilizatorul poate introduce de la consolă comenzi individuale, șiruri de comenzi sau comenzi alternînd cu șiruri de caractere. Parametrii opționali sînt marcați cu paranteze drepte [.].

Comenzile editorului sînt :

A — „Append" : Citește în Zona texte de la cititor ;
B — „Begin" : Începutul Zonei Text ;
C — „Character" : Deplasează indicatorul peste caractere ;
D — „Delete" : Șterge caractere ;
E — „Exit" : Perforează la ieșire textul din Zona ;
F — „Find" : Caută un șir de caractere ;
G — eroare ;
H — eroare ;
I — „Insert" : Introduce un text în Zona Text ;
J — eroare ;
K — „Kill" : Șterge linii de text ;
L — „Line" : Deplasează Indicatorul peste linii ;
M — eroare ;
N — „Null" : Perforează 60 caractere „Null" ;
O — eroare ;
P — eroare ;
Q — eroare ;
R — eroare ;
S — „Substitute" : Înlocuiește un șir de caractere cu altul ;
T — „Type" : Listează la consolă linii din Zona Text ;
U — eroare ;
V — eroare ;
W — „Write" : Perforează un număr de linii din Zona Text ;
X — eroare ;
Y — eroare ;
Z — „End of buffer" : Sfîrșitul Zonei Text.

A — Append. Format : **A$$**

Comanda A este utilizată pentru a introduce în Zona Text, un text citit de la echipamentul asignat ca cititor al sistemuului. Comanda A după ce a fost inițiată, continuă citirea textului pînă la îndeplinirea uneia din următoarele condiții :

— S-a citit End — of-File, care este :

CTRL/Z pentru cititorul de bandă ;

.EOF pentru cititorul de cartele. End-of-File nu se introduce în Zona Text :

— S-a citit caracterul Form-Feed (CTRL/L). Caracterul Form-Feed este introdus în Zona Text ;

— S-au citit 50 linii de text ;

— Zona Text a ajuns la limitele zonei de lucru disponibile (Zonă de lucru plină) ;

— S-a citit pînă la terminarea benzii (nu există End-of-File sau cititorul este oprit în timpul execuției comenzii A) ;

— S-a detectat o eroare la citire. Controlul este redat monitorului.

Prin comanda A se pot citi cel mult 50 linii. Dacă se dorește citirea unui text mai mare, se introduce un șir de comenzi A sau se execută iterativ comanda A, după cum se arată în paragraful următor. Textul citit prin comanda A se introduce la sfîrșitul Zonei Text, imediat după textul deja existent în Zona.

B — Begin. Format : **B$$**

Comanda B este utilizată pentru deplasarea Indicatorului Zonei la începutul Zonei Text (Start Zona).

Cîteva cazuri de utilizare a comenzii B :

— Stabilirea unei referințe pentru numărarea liniilor de text ;

— Deplasarea Indicatorului la începutul Zonei Text pentru căutarea unui șir de caractere sau pentru a lista la consolă conținutul Zonei Text ;

— Inserarea unui text la începutul Zonei.

C — Character. Format : [n]**C$$**

Comanda C necesită un parametru, care se introduce înaintea comenzii și specifică numărul de caractere peste care se va deplasa Indicatorul, număr cuprins între —65535 și +65534. Dacă numărul este pozitiv, Indicatorul din Zona Text se va deplasa spre sfîrșitul acesteia, iar dacă numărul este negativ, spre începutul Zonei Text. Dacă numărul specificat duce la depășirea limitelor Zonei Text, comanda se va termina cînd se ajunge la una din aceste limite.

Această comandă se utilizează, de obicei, pentru a deplasa indicatorul în cadrul aceleiași linii, altfel utilizarea ei devine dificilă.

Dacă parametrul are valoarea 0, comanda nu are nici un efect. Dacă nu este furnizat în comandă, parametrul se consideră implicit egal cu +1.

Delete character. Format : **[n]D$$**

Comanda D necesită un parametru care se introduce înaintea comenzii și specifică numărul de caractere care vor fi șterse din text. Parametrul este un număr zecimal cuprins între —65535 și +65534. Dacă numărul este pozitiv, caracterele vor fi șterse de la poziția Indicatorului

din Zona Text spre sfîrşitul acesteia, iar dacă este negativ, spre începutul Zonei Text. Dacă numărul specificat duce la depăşirea limitelor Zonei Text, comanda se va termina cînd se ajunge la una din aceste limite.

Dacă parametrul are valoarea 0, comanda nu are nici un efect. Dacă nu este furnizat în comandă, parametrul se consideră implicit egal cu +1.

E — Exit. Format : **E$$**

Comanda E este utilizată la sfîrşitul unei sesiuni de lucru, pentru a forma o bandă nouă conţinînd textul editat. Tot conţinutul Zonei Text este trimis la echipamentul asignat ca perforator, iar Zona Text este iniţializată. După ce s-a perforat conţinutul Zonei Text, textul rămas necitit de la READER este citit şi trimis la PUNCH. Apoi se adaugă la sfîrşitul textului caracterul End-of-File (CTRL/Z) şi 60 de caractere Null, ca sfîrşit de bandă.

Trebuie notat că la început, comanda E nu trimite caractere Null pentru a forma un cap de bandă. Dacă Zona Text este vidă, comanda E poate fi utilizată pentru copierea unor benzi de la READER la PUNCH.

După terminarea comenzii E, editorul se reiniţializează şi este pregătit pentru o nouă sesiune de lucru, întocmai ca şi la prima sa intrare în execuţie.

F — Find text string. Format : **F***text***$$**

Comanda F este utilizată pentru căutarea unui şir de caractere din Zona Text. Comanda F aşteaptă un parametru, care se introduce după comandă şi specifică şirul de caractere care trebuie căutat. Parametrul poate conţine cel mult 16 caractere. Dacă şirul este format din mai mult de 16 caractere, nu vor fi luate în considerare decît primele 16. Caracterele ALT MODE,ESC şi CTRL/C nu pot face parte din text, deoarece acestea sînt caractere de comandă.

Comanda F începe căutarea de la valoarea curentă a Indicatorului din Zona Text şi se termină în momentul în care se găseşte un şir de caractere care coincide cu cel specificat ca parametru. Sînt considerate atît caracterele tipăribile cît şi cele netipăribile. În acest caz, la terminarea comenzii, editorul semnalează prin trimiterea caracterului „*“ că aşteaptă o nouă comandă. Poziţia Indicatorului este după ultimul caracter al şirului de caractere găsit. Dacă şirul specificat nu a fost întîlnit pînă la epuizarea Zonei Text, comanda se opreşte în acest moment, Indicatorul rămîne la sfîrşitul Zonei Text şi se tipăreşte mesajul :

CAN NOT FIND “*text*“
BREAK

Editorul aşteaptă apoi o nouă comandă.

Dacă se utilizează un şir de comenzi, textul ataşat comenzii F se va termina cu un caracter ESC, după care urmează alte comenzi. *Este indicat să se verifice terminarea comenzii F prin listarea la consolă a textului găsit.*

I — Insert text. Format : **I***text***$$**

Comanda I se utilizează pentru a introduce în Zona Text, un text de la echipamentul asignat drept consolă. După detectarea caracterului I, editorul intră în execuţia comenzii şi acceptă ca text caracterele introduse de la consolă pînă se detectează unul din următoarele caractere :

— ALT MODE — terminarea șirului de caractere ;
— ESC — terminarea șirului de caractere ;
— CTRL/C — anularea comenzii.

Trebuie notat că la introducerea caracterului CR(0DH), editorul introduce în mod automat caracterul LF(0AH). Deci, dacă se introduce de la tastatură CR, în text va apare perechea de caractere CR, LF. Textul asociat comenzii I este introdus în Zona Text, funcție de poziția Indicatorului. Dacă Zona Text este vidă, șirul de caractere va fi introdus la începutul Zonei. Dacă Zona conține un text iar Indicatorul se găsește la sfîrșitul Zonei Text, șirul de caractere citit de la consolă se va introduce în continuarea textului deja existent în Zona Text (aceasta mărindu-și dimensiunea în mod corespunzător).

Dacă în Zona Text există deja un text iar Indicatorul se găsește undeva într-o poziție intermediară, textul nou va fi introdus începînd cu poziția curentă a Indicatorului Zonei Text, în interiorul textului existent.

K — Kill lines. Format : [n]K$$

Comanda K necesită un parametru, care se introduce înaintea comenzii și specifică numărul de linii care vor fi șterse din text. Parametrul este un număr zecimal cuprins între —65535 și +65534. Dacă numărul este pozitiv, ștergerea liniilor se face de la poziția Indicatorului din Zona Text, spre sfîrșitul acesteia; iar dacă este negativ, spre începutul Zonei Text. Dacă numărul specificat ca parametru duce la depășirea limitelor Zonei Text, comanda se va termina cînd se ajunge la una din aceste limite.

Dacă parametrul are valoarea 0, vor fi șterse caracterele din linia curentă cuprinse între primul caracter din linie și poziția Indicatorului. Dacă parametrul are valoarea 1, vor fi șterse caracterele din linia curentă cuprinse între poziția Indicatorului și caracterul LF, care specifică sfîrșitul liniei, inclusiv caracterul LF. Dacă parametrul lipsește din comandă, se consideră implicit egal cu +1.

L — Line. Format : [n]**L$$**

Comanda L necesită un parametru, care se introduce înaintea comenzii și specifică numărul de linii peste care va fi deplasat Indicatorul din Zona Text. Parametrul poate fi orice număr zecimal cuprins între —65535 și +65534. Dacă numărul este pozitiv, Indicatorul din Zona Text se va deplasa spre sfîrșitul acesteia, iar dacă este negativ, spre începutul Zonei Text. Dacă numărul specificat duce la depășirea limitelor Zonei Text, comanda se va termina cînd se ajunge la una din aceste limite.

Dacă parametrul are valoarea 0, Indicatorul din Zona Text va fi poziționat la începutul liniei curente (înaintea primului caracter din linie). Dacă parametrul are valoarea —1, Indicatorul va fi poziționat la începutul liniei care precede linia curentă. Dacă parametrul are valoarea +1, Indicatorul din Zona Text va fi poziționat la sfîrșitul liniei curente (după caracterul LF care delimitează această linie). Dacă parametrul lipsește din comandă, se consideră implicit egal cu +1. Dacă se specifică în comandă doar semnul —, valoarea parametrului se consideră —1.

N — Null. Format : **N$$**

Comanda N este utilizată pentru a perfora început sau sfîrşit de bandă. Fiecare comandă N perforează 60 de caractere Null.

S — Substitute text string. Format : **S** *şir vechi* [**$** *şir nou*]**$$**

Comanda S este utilizată pentru căutarea unui şir de caractere şi înlocuirea acestuia (dacă este găsit), cu un alt şir de caractere. Partea de căutare a unui şir de caractere este similară cu comanda F. Partea de înlocuire cu un alt şir de caractere se execută numai dacă şirul căutat a fost găsit. Şirul nou, care înlocuieşte şirul căutat, poate avea oricîte caractere (excluzînd caracterele ALT MODE, ESC şi CTRL/C). La terminarea comenzii, Indicatorul din Zona Text se găseşte după ultimul caracter din şirul nou introdus.

Dacă şirul vechi nu a fost găsit, comanda se termină cînd căutarea ajunge la sfîrşitul Zonei Text, iar Indicatorul se va găsi la sfîrşitul Zonei Text. Dacă în comanda S nu se specifică şirul nou, înlocuitor, şirul căutat (şirul vechi), dacă este găsit, este şters. În acest fel se pot căuta şi şterge selectiv şiruri cu mai puţin de 16 caractere.

T — Type out text. Format : **[n]T$$**

Comanda T necesită un parametru, care se introduce înaintea comenzii şi specifică numărul de linii care vor fi listate la consolă. Parametrul este un număr zecimal cuprins între —65535 şi +65534. Dacă numărul este pozitiv, listarea liniilor începe cu poziţia curentă a Indicatorului şi continuă pînă ce numărul de linii afişate (inclusiv linia curentă) este cel specificat în comandă. Dacă numărul este negativ, afişarea începe cu linia dată de valoarea curentă a Indicatorului din Zona Text, minus numărul specificat ca parametru şi continuă pînă la poziţia curentă a Indicatorului. Dacă numărul specificat duce la depăşirea limitelor Zonei Text, comanda se va termina cînd se ajunge la una din aceste limite.

Dacă parametrul are valoarea 0, se listează la consolă caracterele de la începutul liniei curente pînă la poziţia curentă a Indicatorului din Zona Text. Dacă parametrul are valoarea 1, se listează caracterele începînd cu poziţia Indicatorului pînă la sfîrşitul liniei curente (determinat de caracterul LF). Dacă parametrul lipseşte din comandă, se consideră implicit egal cu +1.

Comanda T nu modifică valoarea Indicatorului din Zona Text.

W — Write. Format : **nW$$**

Comanda W necesită un parametru, care se introduce înaintea comenzii şi specifică numărul de linii care vor fi perforate la dispozitivul asignat ca PUNCH.

Parametrul este un număr zecimal cuprins între —65535 şi +65534. *Indiferent de semnul numărului comanda W consideră parametrul ca număr pozitiv. Indiferent de poziţia curentă a Indicatorului, perforarea se efectuează începînd cu prima linie din Zona Text.*

Pe măsură ce se perforează o linie, aceasta este ştearsă şi textul rămas se compactează la începutul Zonei Text. Indicatorul se poziţionează la începutul Zonei Text.

Dacă parametrul are valoare 0, nu se va perfora nici o linie.

Comanda W se utilizează pentru eliberarea Zonei Text atunci cînd se creează sau se editează texte de dimensiuni mari, ce depăşesc capacitatea Zonei Text.

Z — End of buffer. Format : **Z$$**

Comanda Z se utilizează pentru a deplasa Indicatorul la sfîrşitul Zonei Text, după ultimul caracter din Zonă.

7.1.3. Caractere speciale şi utilizarea lor

Corectarea erorilor de tipărire

Erorile de tipărire într-un şir de comenzi sau de caractere, pot fi înlăturate prin tastarea caracterului RUBOUT(7FH), pentru fiecare caracter ce trebuie înlăturat. La fiecare caracter şters, începînd cu ultimul caracter introdus, editorul trimite în ecou caracterul şters. Utilizarea caracterului RUBOUT pentru corectarea erorilor de tipărire, este posibilă numai înaintea introducerii terminatorului de comandă sau şir de comenzi („ESC“ „ESC“).

Anularea comenzilor

Dacă se doreşte anularea unui şir de comenzi, înainte de introducerea terminatorului de comenzi, se introduce caracterul CTRL/C(03H). Editorul anulează tot şirul de comenzi şi trimite caracterul „*“ pentru a semnaliza aşteptarea unei noi comenzi.

De asemenea, dacă în timpul execuţiei comenzilor se introduce caracterul CTRL/C, editorul suspendă execuţia comenzilor, şi trece în bucla de aşteptare a unei noi comenzi. Funcţie de consola utilizată, acelaşi efect se obţine acţionînd tasta „BREAK“. Indiferent în ce stare ar fi, editorul revine în bucla de aşteptare a unei noi comenzi. Indicatorul din Zona Text rămîne la valoarea pe care o avea în momentul abandonării comenzii.

Tabulare

Caracterul de tabulare orizontală — CTRL/I(09H) poate fi utilizat pentru editarea textelor cu ETX18. Trebuie notat că acest caracter este introdus în Zona Text ori de cîte ori este tastat la consolă. La întîlnirea caracterului CTRL/I, editorul generează un număr suficient de spaţii (20H) pentru a deplasa carul consolei (sau cursorul) la următoarea poziţie de tabulare. Aceste poziţii sînt fixe, la fiecare opt caractere, în cadrul unei linii de text.

Caracterele CR (0DH) şi LF(0AH)

Caracterele CARRIAGE RETURN şi LINE FEED, sînt introduse în Zona Text ca şi caractere obişnuite. Totuşi, trebuie notat că la utilizarea comenzii Insert, introducerea de la consolă a caracterului CR are ca efect memorarea în Zonă a perechii de caractere CRLF.

Dacă textul se prepară „*off line*“ şi se va introduce în Zona Text cu comanda **Append,** aceasta nu introduce nici un caracter în plus, deci pe bandă trebuie să apară atît CR cît şi LF.

7.1.4. Modul de utilizare a editorului ETX18

În general, secvența de creare a unui fișier sursă nou constă din pornirea editorului, introducerea de la consolă a textului, editarea acestuia și formarea unei benzi cu fișierul sursă editat.

Pentru a edita un fișier deja existent se pornește editorul, se citește fișierul sursă de pe bandă, se editează și se formează o bandă cu noua versiune (editată) a textului. Informațiile trimise de utilizator editorului pot semnifica o comandă, un șir de comenzi, un șir de caractere reprezentînd un text sau caractere de comandă. Ori de cîte ori editorul așteaptă o nouă comandă, trimite la consolă, la început de linie, caracterul „*". După introducerea unei comenzi sau a unui șir de comenzi, utilizatorul trebuie să introducă de la consolă un terminator constînd din două caractere ALT MODE(7DH) sau ESC(1BH). La detectarea acestor caractere, editorul trimite în ecou caracterele „$$" și trece în regim de execuție a comenzilor.

Pentru a specifica terminarea unui text (asociat comenzilor F, I, S) se utilizează un caracter ALT MODE sau ESC (editorul trimite în ecou „$"). Dacă după acest șir de caractere nu mai urmează nici o comandă, se introduce terminatorul de comenzi-două caractere ALT MODE sau ESC, subînțelegîndu-se în acest caz și terminarea șirului de caractere.

Toate operațiile de I/E se efectuează prin monitor, deci înainte de lansarea în execuție a editorului trebuie făcută asignarea configurației de I/E dorită.

Utilizarea comenzilor B, Z, I, A, T

Să presupunem că Zona Text este goală iar pe o bandă se găsește perforat următorul text :

L2 ACEASTA ESTE LINIA 2
L3 : ACEASTA ESTE LINIA 3
L4 ACEASTA ESTE LINIA 4
L5 ACEASTA ESTE LINIA 5

Citirea textului, introducerea lui în Zona Text și listarea la consolă se pot efectua cu șirul de comenzi :

***A4T$$**

Dacă textul ce trebuie introdus de pe bandă are mai mult de 50 de linii, se utilizează un șir de comenzi : ***AAA$$** (pentru citirea unui text de 150 linii).

Pentru introducerea și listarea la consolă a unei noi linii, înaintea liniei 2, se vor utiliza comenzile :

*IL1 ACEASTA ESTE LINIA 1
***-1T$$**
L1 ACEASTA ESTE LINIA 1
*

După L1 s-a introdus caracterul de tabulare CTRL/I, care este netipăribil, de asemenea după caracterele „LINIA 1" s-a introdus caracterul CR, fiind adăugat de editor și caracterul LF.

Poziția curentă a indicatorului din Zona Text poate fi găsită cu comanda :

***T$$**
L2 ACEASTA ESTE LINIA 2

Deci, Indicatorul se găsește între caracterele LF din prima linie și L din linia a doua.

Listarea la consolă a ultimei linii se poate realiza cu șirul de comenzi

***Z-1T$$**
L5 ACEASTA ESTE LINIA 5
*

Dacă se dorește listarea la consolă a întregului text, dar nu se știe exact numărul de linii și poziția curentă a Indicatorului se utilizează șirul de comenzi :

***B20T$$** Am considerat că textul este format din mai puțin de 20 de linii.

Utilizarea comenzilor C, D, L, K

Considerăm că nu se cunoaște poziția curentă a Indicatorului și dorim să ștergem caracterul „:“ din linia 3. Se va utiliza următorul șir de comenzi :

***B2L2CD0TT$$**
L3 ACEASTA ESTE LINIA 3
*

De remarcat că listarea la consolă a liniei curente, poziția indicatorului fiind undeva în linia respectivă, se poate face cu comenzile :

***0TT$$** Prin comanda 0T se afișează caracterele din linie pînă la poziția Indicatorului, iar prin comanda T cu argumentul 1 (implicit) caracterele de la poziția curentă a Indicatorului pînă la sfîrșitul liniei. Indicatorul din zona text se găsește în linia 3, între caracterele L și CTRL/I și dorim să ștergem cuvîntul ACEASTA.

***C8D0TT$$**
L3 ESTE LINIA 3
*

Pentru a șterge liniile 2, 3, 4 și 5 trebuie poziționat Indicatorul la începutul liniei 2 și apoi trebuie șterse 4 linii.

***-1L4K$$**

Se poate utiliza și o altă combinație de comenzi.

***L-2K2K$$** Comanda L poziționează Indicatorul la sfîrșitul liniei 3, apoi se șterg liniile 3 și 2 cu comanda —2K și liniile 4, 5 cu comanda 2K.

Utilizarea comenzilor F, S

Considerăm textul format din liniile L1, L2, L3, L4 și L5.

Ștergerea caracterului „:“ din linia 3 se poate face cu comenzile :

***BFL3 : $-1D0T$**
L3* sau
***BFL3$D$$** sau
*BSL3 : $L3$0TT$$
L3 ACEASTA ESTE LINIA 3
sau
***BS : $$**

Dacă acum se introduce comanda de căutare a șirului L3, căutarea va începe de la poziția curentă a Indicatorului pînă la sfîrșitul Zona Text. L3 nu va fi găsit, fapt semnalizat de editor printr-un mesaj.

***FL3$$**
CAN NOT FIND „L3“
BREAK

Utilizarea comenzilor W, E, N

Dacă textul ce trebuie editat depăşeşte dimensiunea zonei de lucru disponibile, editarea se va face pe porţiuni.

La terminarea editării unei porţiuni de text, versiunea editată se trimite la PUNCH cu comanda W. Dacă sînt aproximativ 150 linii de text, se poate utiliza comanda :

***200W$$.** Poziţia Indicatorului nu are importanţă. Perforarea începe întotdeauna cu prima linie.

Dacă Zona Text conţine ultima porţiune editată dintr-un text, perforarea se va face cu comanda :

***E$$.** La sfîrşitul textului se perforează caracterul de sfîrşit de fişier, CTRL/Z şi 60 de caractere NULL.

Dacă editarea nu s-a terminat, dar trebuie întreruptă, se poate utiliza şirul de comenzi :

***NNE$$**

După executarea comenzii E editorul se reiniţializează, şi trece în bucla de aşteptare comenzi.

Executarea iterativă a comenzilor

O comandă sau un şir de comenzi pot fi repetate de un număr oarecare de ori dacă sînt incluse între paranteze unghiulare „<“ şi „>“ şi se specifică numărul de repetări.

Formatul comenzilor cu repetare este următorul :

***n**<*comandă sau şir de comenzi*>**$$**

n specifică numărul de repetări a comenzilor înscrise între parantezele „<“ şi „>“. Parantezele unghiulare apar efectiv în linia de comenzi. De exemplu, dacă în textul format din cele cinci linii, utilizat anterior pentru exemplificări, se doreşte înlocuirea caracterelor L1 cu 1, L2 cu 2 etc.

Se poate utiliza, în mod repetat comanda S :

***BSL1L10<S**
L$
>$$
CAN NOT FIND „L“
*****

Să presupunem că avem un tabel cu puterea a treia a numerelor întregi cuprinse între 1400 şi 1500 :

```
1400 2744000000
1401 2743884201
1402 2755776808
1403 2761677827
. . . . . . . . . . . . . . .
. . . . . . . . . . . . . . .
1438 3361517992
1499 3368254499
1500 3375000000
```

Dacă dorim să scriem acest tabel într-o formă mai uşor de citit se poate utiliza succesiunea de comenzi :

***B101<6CI $2<3CI $>L>$$**

Tabelul de mai sus o să apară în următorul format :

1400 2 744 000 000
1401 2 749 884 201
1402 2 755 776 808
1403 2 761 677 827
. . . .
. . . .
1498 3 361 517 992
1499 3 368 254 499
1500 3 375 000 000

Utilizarea iterativă a comenzilor este permisă pînă la maxim opt nivele de adîncime.

7.1.5. Mesajele editorului către utilizator

La intrarea în execuţie editorul trimite la consolă mesajul :

EDITOR DE TEXTE ETX18 V x.x

De asemenea, după executarea comenzii E, editorul se reiniţializează şi afişează la consolă mesajul de mai sus.

Dacă editorul primeşte un caracter neaşteptat în faza de execuţie respectivă, trimite la consolă mesajul :

„c“ ILLEGAL IN THIS CONTEXT

„c“ este caracterul inadecvat, introdus de la consolă (de exemplu, cînd se aşteaptă o comandă, se introduce un caracter alfanumeric care nu este identificator pentru o comandă legală).

Dacă la execuţia comenzilor F,S, şirul căutat n-a fost găsit pe porţiunea de text dintre valoarea curentă a Indicatorului şi sfîrşitului Zonei Text, editorul emite mesajul :

CAN NOT FIND *„şir caractere“*

BREAK

„şir caractere“ reprezintă şirul căutat dar negăsit. Comanda este abandonată, fapt semnalizat prin mesajul :

BREAK

Dacă la utilizarea iterativă a comenzilor nivelul de adîncime este mai mare decît opt, editorul emite mesajul :

ITERATION STACK FAULT

BREAK

Comanda este abandonată şi se revine în regim de aşteptare comenzi.

7.1.6. Încărcarea şi lansarea în execuţie a editorului

Editorul de texte este rezident pe bandă perforată, bandă magnetică sau cartele perforate.

Dacă editorul este rezident pe bandă perforată se procedează astfel :

— Se lansează monitorul, dacă acesta nu este deja în execuţie, după procedura descrisă în capitolul 3 ;

— Se pregăteşte cititorul de bandă, se introduce banda în cititor ;

— Se asignează cititorul de bandă ca cititor al sistemului (dacă este cazul) :

.AR = P <CR> ;

— Se citeşte în memorie editorul :

.R <CR> ;

— După terminarea operaţiei de citire a benzii, se fac asignările dorite pentru lucrul cu editorul. Dacă se lucrează cu casetele, se fac deschiderile de fişiere pentru citire/scriere, după caz ;

— Se lansează în execuţie editorul :

.G40 <CR> ;

Editorul va răspunde la consolă cu mesajul :

EDITOR DE TEXTE ETX18 V x.x

*

Dacă editorul este rezident pe casetă sau bandă magnetică, se procedează astfel :

— Se asignează ca cititor una din unităţile de casetă sau bandă magnetică :

.AR = K sau **.AR** = M funcţie de unitatea care conţine Editorul-program obiect ;

— Se deschide pentru citire fişierul cu numărul „n" pe unitatea „m" care conţine editorul sub formă de program obiect format hexa-ASCII :

.KIm,n <CR>Monitorul va căuta pe unitatea „m" fişierul „n" ;

— Se citeşte în memorie editorul :

.R <CR> ;

— Se închide fişierul de intrare (nu este obligatorie) :

.KC <CR> ;

— În continuare se procedează ca în cazul în care editorul este rezident pe bandă perforată.

Revenirea în Monitor, fie la sfîrşitul unei sesiuni de lucru fie pentru a schimba asignările de periferice, se face prin întrerupere de la panoul frontal. Dacă trecerea în monitor a fost temporară, revenirea în editor se realizează cu comanda :

.G <CR>

La sfîrşitul unei sesiuni de editare în care s-au utilizat casetele sau benzile magnetice, este necesar ca la revenirea în Monitor să se închidă fişierele de ieşire.

Dacă din cauza unor incidente (căderi instantanee ale reţelei etc.), editorul nu mai funcţionează corect, se va reîncărca şi lansa în execuţie editorul.

7.2. Editorul de texte EDIT

Editorul de texte EDIT este o variantă a editorului ETX18 care se execută sub SFDX-18. Astfel EDIT poate fi utilizat pe calculatoarele M18, M18B sau M118 ce au în configurație cel puțin o unitate de discuri flexibile. Diferențele față de ETX18 se referă la :

— setul de comenzi ;

— utilizarea editorului EDIT.

7.2.1. Comenzile editorului EDIT

• Față de comenzile editorului ETX18, EDIT a fost prevăzut cu următoarele două comenzi suplimentare :

Q—Quit. Format : **Q$$**

Comanda Q specifică terminarea editării, reîntoarcerea controlului în SFDX-18 fără salvarea textului din Zona Text. Dacă fișierul în curs de editare este un fișier nou creat, datele introduse nu sînt salvate și numele fișierului este șters de pe disc. Dacă însă fișierul în curs de editare este un fișier vechi, creat anterior, prin execuția comenzii Q fișierul rămîne intact, toate modificările efectuate în memorie sînt ignorate.

M—MEMORY. Format : **M$$**

Comanda M calculează și afișează la consola sistemului numărul de octeți disponibili în Zona Text, sub forma unui mesaj cu structura următoare :

nnn—**CHARACTER(S) AVAILABLE IN WORKSPACE**

unde *nnn* este un număr întreg zecimal reprezentînd informația dorită.

Editorul EDIT nu recunoaște comanda N-Null. De asemenea la execuția comenzilor care implică un transfer de date cu dispozitivele de I/E, aceasta se face sub controlul SFDX-18. Fișierele de intrare și ieșire sînt specificate la lansarea în execuție a editorului.

După execuția comenzii E-Exit control revine la SFDX-18.

7.2.2. Utilizarea editorului EDIT

Editorul trimite la consolă caracterul „*" atunci cînd este gata să accepte o comandă.

Edit interpretează caracterele de control conform cu convențiile din SFDX. Astfel, în plus față de caracterele speciale recunoscute de ETX18, EDIT recunoaște și interpretează următoarele caractere de control :

— CTRL/R — Tipărește linia de comandă în curs de introducere, în întregime și corecturile făcute ;

— CTRL/X — Anulează linia de comandă curentă, se tipărește la consolă caracterul (#) și se așteaptă introducerea unei noi linii de comenzi.

Trebuie notat că interpretarea caracterului de tabulare, CTRL/I la editarea programelor sursă în PL/M se face diferit de către PL/M și EDIT.

Astfel CTRL/I avansează carul (cursorul) cu opt poziţii pentru EDIT şi cu patru pentru PL/M.

Lansarea în execuţie a editorului se face cu o comandă SFDX de forma :

EDIT *fişier sursă* [**TO** *fişier destinaţie*] <*CR*> unde *fişier sursă* este un fişier ce urmează să fie creat pe disc şi editat sau este un fişier deja creat care urmează să fie actualizat. În acest caz, *fişier sursă* poate fi orice fişier de intrare recunoscut de SFDX.

Dacă *fişier sursă* nu este un fişier pe disc, *fişier destinaţie* trebuie să fie specificat şi trebuie să fie un fişier pe disc.

Dacă *fişier sursă* este pe disc şi *fişier destinaţie* nu este specificat, pot apare următoarele situaţii :

— *fişier sursă* este un fişier nou. SFDX creează şi deschide pentru scriere acest fişier ;

— *fişier sursă* este un fişier existent şi nu este protejat la scriere. SFDX deschide acest fişier pentru citire şi creează, pentru ieşire un fişier temporar cu numele EDIT.TMP. La execuţia comenzii E-Exit cele două fişiere sînt închise şi redenumite astfel că *fişier sursă* devine *nume* .BAK iar EDIT.TMP devine *fişier sursă ;*

— *fişier sursă* este un fişier existent şi protejat la scriere. Controlul revine în SFDX şi se emite un mesaj de eroare la consola sistemului.

7.3. CREDIT — Editor de texte orientat pe ecran

7.3.1. Prezentarea generală a editorului

CREDIT reprezintă un editor de texte orientat pe dispozitive de afişare cu tub catodic, compatibil cu sistemul de operare SFDX-18. CREDIT se execută pe microcalculatoarele M118 şi M18 dotate cu terminal cu tub catodic. Operaţiile de introducere iniţială a unui text şi operaţiile de modificare a unui text deja introdus sînt afişate direct pe ecranul terminalului.

Prin funcţiile sale, CREDIT oferă facilităţi avansate de editare a textelor formate din şiruri de caractere în cod ASCII, dar este în principal orientat pe editarea textelor reprezentînd programe sursă. Principalele funcţii ale editorului sînt :

Editare video : O parte din textul introdus sau corectat este afişat pe ecranul terminalului astfel că operaţiile de editare sînt observate imediat pe ecran. Utilizînd funcţiile ce acţionează în mod ecran, cum ar fi : poziţionarea cursorului, ştergerea sau adăugarea de caractere sau porţiuni de text, se pot edita cu uşurinţă texte, existînd o reacţie vizuală directă a operaţiilor executate.

Editare prin comenzi : În plus faţă de posibilităţile de editare video, CREDIT dispune de un set puternic de comenzi care permit efectuarea unor operaţii de editare mai complexe cum ar fi : găsirea unui şir de

caractere dat și ștergerea acestuia sau înlocuirea cu alt șir, mutarea unei porțiuni de text dintr-o zonă în alta etc. În mod comandă CREDIT oferă funcții de editare similare cu cele ale editoarelor ETX18 sau EDIT.

Comenzi avansate de editare: CREDIT este prevăzut cu o serie de facilități avansate de editare ce permit lucrul direct cu fișiere pe disc, definirea și utilizarea macrocomenzilor, configurarea editorului în funcție de caracteristicile terminalului utilizat.

Prin definirea și apelarea unor macrocomenzi utilizatorul poate dezvolta un set de comenzi specifice unui anumit gen de texte și operații cu acestea.

Gestiunea spațiului pe disc: Dimensiunea textului editat nu este limitată de dimensiunea memoriei RAM ci numai de spațiul disponibil pe discul ce conține textul. CREDIT gestionează spațiul pe disc, aduce în memorie o parte din text ce conține porțiunea în curs de editare fără precauții din partea utilizatorului.

Afișarea modului de utilizare. Editorul are asociat un fișier ce conține o scurtă prezentare a modului de utilizare și a formatului comenzilor. Astfel, în mod comandă, prin introducerea de la tastatură a comenzii H(Help) se afișează, pagină cu pagină, acest fișier. Adaptarea editorului la un terminal oarecare trebuie însoțită de modificarea corespunzătoare, în mod explicit, a fișierului CREDIT.HLP ce conține acest text.

Modul de lucru: Deși CREDIT poate fi utilizat în mod comandă cu un terminal oarecare, eficiența lui constă în utilizarea facilităților unui terminal cu afișare pe ecran. Ecranul terminalului poate să aibă 25 sau 24 de linii dintre care primele 5, respectiv 4, formează Zona Comenzi iar celelalte 20 formează Zona Text. Comenzile introduse în *„mod comandă"* sînt afișate în Zona Comenzi. Dacă se introduc mai mult de 5(4) linii în mod comandă, Zona Comenzi este expandată pe tot spațiul ecranului iar la revenire în *„mod ecran"*, Zona Text este refăcută pe ecran. Cele două zone sînt despărțite de 5 caractere „-".

O linie de text este formată dintr-un șir de caractere alfanumerice terminat cu caracterele CR și LF. La introducerea de la tastatură a caracterului CR editorul introduce în fișierul editat perechea CR, LF și afișează pe ecran un semn special (de obicei ↑) ca indicator de terminator de linie. Lungimea unei linii poate fi oarecare. Pentru a putea introduce linii care depășesc dimensiunea ecranului (de obicei 80 caractere) este necesar ca Terminalul să treacă implicit la linia următoare la depășirea lungimii liniei. Caracterele netipăribile nu sînt afișate în mod ecran și deci nu pot fi referite în nici un fel iar în mod comandă acestea sînt afișate ca și caractere de CONTROL fiind afișate în formatul ↑ C unde C reprezintă caracterul care corespunde caracterului netipăribil CTRL/C (C este un caracter ASCII oarecare).

Pentru a specifica poziția în text asupra căreia acționează comenzile de editare, editorul menține un Indicator care marchează un caracter din fișierul editat. Toate modificările textului se fac relativ la poziția acestuia. Indicatorul este marcat pe ecran cu un caracter special (|, ↑, /). Pe lîngă acest indicator există cursorul terminalului care marchează caracterul curent. În mod ecran, cursorul indică, în general, aceeași poziție ca și indicatorul spre deosebire de lucrul în mod comandă unde nu există această asociere. Cursorul indică doar locul unde se afișează liniile de

comandă introduse. Cursorul poate fi deplasat pe ecran în cele patru direcții prin introducerea unor caractere de control funcție de tastatura terminalului (Ex : ←, , →, ↑ ↓). Încercarea de deplasare a cursorului înafara limitelor ecranului este semnalizată sonor (bell). Poziția cursorului corespunde cu cea a Indicatorului numai dacă cursorul este poziționat pe un caracter.

Nu încercați introducerea unor comenzi de modificare a textului în mod ecran, dacă cursorul nu este poziționat pe un caracter vizibil pe ecran sau pe caracterul ce arată poziția Indicatorului. La unele terminale Indicatorul din zona text și terminatorul de linie sînt afișate cu același semn (↑), ceea ce poate crea confuzii ducînd la „pierderea" poziției indicatorului. În acest caz se poate proceda astfel :

— Se trece în mod comandă ;
— Se modifică cu comanda ALTER semnul ce reprezintă terminatorul de linie ;
— Se aduce indicatorul la începutul (JTT) sau la sfîrșitul (JTE) fișierului în curs de editare ;
— Se deplasează indicatorul în poziția dorită ;
— Se trece în mod ecran (CTRL/V).

De asemenea editorul definește și utilizează 4 marcaje permanente dintre care două indică începutul, respectiv sfîrșitul fișierului în lucru. Dacă textul editat nu încape în memoria disponibilă, se pun marcaje și la începutul și la sfîrșitul zonei de text ce încape în memorie la un moment dat. Cele 4 marcaje permanente menținute de editor sînt denumite :

— TT — (Top of Text) marchează începutul fișierului ;
— TE — (End of file) marchează sfîrșitul fișierului ;
— TB — marchează începutul porțiunii de fișier din memorie ;
— TZ — marchează sfîrșitul porțiunii de fișier din memorie.

Marcajele denumite „tags" reprezintă o modalitate eficientă și comodă de a memora diferite poziții în fișierul editat și deplasarea cursorului în timpul editării la oricare din aceste poziții. Aceste marcaje nu sînt introduse în textul editat și deci ele au o valabilitate temporară doar pe durata unei sesiuni de editare.

În plus față de cele 4 marcaje menținute de editor, utilizatorul poate defini prin comanda TS (tagset) pînă la 10 marcaje numerotate de la 0 la 9. Inițial toate aceste marcaje sînt nesetate și orice încercare de deplasare a cursorului la un astfel de marcaj va da eroare. Marcajele sînt foarte utile în prelucrarea, în timpul editării, a unor blocuri din text.

7.3.2. **Utilizarea editorului-CREDIT**

CREDIT se execută sub sistemul de operare SFDX, ca o comandă SFDX. Fișierul ce urmează să fie editat poate fi un fișier nou sau unul deja existent. La terminarea unei sesiuni de editare se poate înlocui vechiul fișier (dacă exista) cu fișierul editat, fie se păstrează ambele versiuni sub nume diferite. Dacă la un moment dat au fost comise greșeli care compromit sesiunea de editare se poate reveni în SFDX abandonînd versiunea ce era în curs de editare și păstrînd vechea versiune. Fișierul

de intrare poate fi un fişier pe disc sau orice echipament de tip „cititor" recunoscut de SFDX.

Lansarea în execuţie a editorului

Deoarece editorul este format dintr-un singur segment, în întregime rezident în memorie, CREDIT poate fi încărcat şi lansat în execuţie prin introducerea numelui şi a parametrilor, sau poate fi încărcat cu DEBUG şi lansat în execuţie prin MONITOR dacă, de exemplu, trebuie schimbat discul care conţine CREDIT în format obiect cu un disc de lucru ce va conţine fişierul editat. Acest lucru nu este posibil dacă se utilizează un fişier de comenzi, după cum se va arăta în continuare.

Formatul comenzii de lansare în execuţie a editorului este : [:**Fi**:] **CREDIT** *nume fişier 1* [**TO** *nume fişier 2*] [**MACRO** [*(fişier comenzi)*]] <cr>

sau :

DEBUG [:Fi:] **CREDIT** *nume fişier 1* [**TO** *nume fişier 2*] *<cr>*

Se poate utiliza şi în acest caz un fişier de comenzi dar trebuie avut grijă ca la lansarea în execuţie a editorului fişierul cu comenzile să fie pe disc conform cu specificaţia din comandă.

Fi cu i = 0, . . . , 3 specifică unitatea de disc ce conţine programul obiect cu numele CREDIT.

nume fişier 1 poate fi orice echipament de intrare sau nume de fişier pe disc recunoscut de SFDX.

Formatul unui nume de fişier este :

[*:echipament:*] [*nume*] [*.extensie*]] unde :

:echipament: — este orice combinaţie de 2 caractere recunoscută de SFDX ca nume de echipament de I/E valid. Dacă echipamentul specificat este altceva decît discul, *nume. extensie* sînt ignorate. *Dacă nu se specifică nimic în acest cîmp, se consideră implicit :***F0***:*

nume — este un şir de cel mult 6 caractere. *Nume* este obligatoriu dacă *echipament* este discul. Dacă nu, *nume.extensie* sînt ignorate.

extensie — este un şir de cel mult 3 caractere, este opţional dar dacă este specificat trebuie reparat prin punct (.) de *nume*.

TO *nume fişier 2* — specifică faptul că versiunea modificată a fişierului va fi memorată cu numele respectiv. Versiunea veche cu numele *nume fişier 1* rămîne neschimbată. Dacă nu se specifică în comandă cîmpul **TO** *nume fişier 2*, versiunea nouă a fişierului o va înlocui pe cea veche, dar se păstrează totuşi şi versiunea veche sub numele *nume*. **BAK**.

MACRO [*(fişier comenzi)*] — specifică încărcarea unui fişier ce conţine comenzi ale editorului. Numele fişierului de comenzi este opţional şi dacă lipseşte se consideră implicit fişierul cu numele *:***F0***:* **CREDIT.MAC**

Deci pentru lansarea în execuţie a editorului trebuie ca SFDX să fie în regim de aşteptare comenzi, unul din discurile sistemului să conţină comanda CREDIT şi eventual fişierul de comenzi utilizat, fişierul specificat ca destinaţie, sau fişierul sursă dacă nu se specifică destinaţia iar *discul pe care se găsesc acestea să nu fie protejate la scriere*. La începutul sesiunii de editare, CREDIT afişează pe ecran un mesaj de lansare în execuţie şi un mesaj ce indică spaţiul disponibil pentru editare.

Dacă fişierul specificat în comandă există deja, apare următorul mesaj :

OLD FILE SIZE = *mm* **BLKS LEFT** = *nnnn* unde :

mm — reprezintă numărul de blocuri ce formează vechiul fişier ;

nnnn — numărul blocurilor de pe disc nefolosite.

Dacă fişierul specificat pentru editare este un fişier nou, CREDIT afişează următorul mesaj :

NEW FILE BLKS LEFT = *nnnn*

Dacă apare o eroare în secvenţa de lansare a editorului, se afişează un mesaj corespunzător şi controlul revine în SFDX. Dacă lansarea se termină cu succes, CREDIT intră în execuţie în mod ecran. Utilizatorul trebuie să ştie cel puţin comanda de trecere în mod comandă (de obicei SHIFT/CTRL/M) care depinde de tastatura utilizată, apoi prin comanda Help(H) se poate iniţia în utilizarea comenzilor editorului.

Dacă tastatura şi funcţiile ecranului nu corespund cu cele considerate implicit în CREDIT şi deci se utilizează un fişier MACRO pentru a configura editorul conform cu caracteristicile terminalului, la lansarea în execuţie comenzile de ştergere ecran şi poziţionare a cursorului în poziţia iniţială pot să apară pe ecran sub forma unor *„mesaje ciudate"*. După configurarea specificată în fişierul de macrocomenzi editorul va funcţiona corect.

Revenirea din editor

Există trei modalităţi de terminare a unei sesiuni de editare :

— Se înlocuieşte vechea versiune a fişierului cu versiunea modificată ;

— Se memorează versiunea modificată cu un nume diferit ;

— Se ignoră toate modificările făcute şi se lasă nemodificat vechiul fişier.

Comanda EX : are ca efect terminarea sesiunii de editare şi memorarea versiunii modificate sub un nume nou prin înlocuirea vechii versiuni.

Formatul comenzii este :

EX [*nume fişier*] <cr> unde **EX** este numele comenzii iar *nume fişier* reprezintă numele sub care va fi memorată versiunea modificată. Dacă nu se specifică nume fişier, memorarea se va face conform comenzii de lansare în execuţie. Dacă se introduce de la tastatură EXIT, editorul va memora versiunea modificată sub numele **:F0:IT** şi va păstra nemodificat fişierul vechi.

Comanda EQ : are ca efect terminarea sesiunii de editare cu ignorarea tuturor modificărilor făcute. Pentru a reduce posibilitatea unei greşeli de operare editorul întreabă dacă utilizatorul este conştient de urmarea acestei comenzi prin mesajul :

QUIT ?

Orice răspuns diferit de Y sau y are ca efect continuarea editării în curs.

Formatul comenzii este : **EQ**<cr>

Fişierul reprezintă singura modalitate de a memora, cu caracter permanent, datele în calculator. Zona text din memorie sau fişierele temporare menţinute de CREDIT în timpul unei editări reprezintă forme nesigure de memorare a textului editat.

— *Nu uitați să salvați într-un fișier textul editat la sfîrșitul sesiunii de editare ;*

— *Înainte de introducerea comenzii EX verificați dacă există condițiile de efectuare a acestei comenzi (De exemplu există disc în unitatea specificată și ușa este închisă etc.) ;*

— Pentru a preveni compromiterea din diferite motive a unei sesiuni de editare complexe, *se recomandă ca periodic să se salveze textul editat și să se reînceapă o nouă sesiune de editare pentru a continua ;*

— Dacă în timpul editării în mod ecran *„se pierde"* Indicatorul prin deplasarea inadecvată a cursorului, *nu vă impacientați.* Greșeala este remediabilă. *Citiți cu atenție Modul de lucru.*

7.3.3. Editarea în mod ecran

În acest paragraf se va prezenta modul în care se poate introduce și modifica un text afișat pe ecran în Zona Text utilizînd un set de reguli restrîns, dar care acoperă în mare parte cerințele de introducere și modificare a unui text. Pentru modificări mai complexe se pot utiliza funcțiile de editare în mod comandă care vor fi prezentate în paragraful următor. Avantajul editării în mod ecran constă în faptul că rezultatele comenzilor se văd imediat pe ecran facilitînd astfel editarea și evitarea distrugerii unor porțiuni de text prin comenzi eronate. La lansarea în execuție, editorul trece în mod ecran. Trecerea din mod comandă în mod ecran se face cu comanda CTRL/V iar trecerea din mod ecran în mod comandă se face cu comanda Home sau SHIFT/CTRL/M, depinde de tastatura utilizată.

În mod ecran, CREDIT avertizează sonor utilizatorul în următoarele situații :

— Cursorul a ajuns în poziția 70 a unei linii (considerată de 80 caractere) ;

— La introducerea unei comenzi ce nu este acceptată de editor (ex. deplasarea cursorului în afara limitelor ecranului, schimbarea textului deși cursorul nu este poziționat pe text).

Pentru a introduce un text trebuie doar să tastați caracterele respective. Caracterele ce nu sînt caractere de comandă vor fi introduse în text. Caracterul (tipăribil) introdus de la tastatură este imediat afișat pe ecran iar cursorul și Indicatorul se deplasează pe poziția următoare. Pentru a introduce în text un caracter de control, acesta va fi precedat de (\), utilizat ca și caracter de literalizare. Astfel pentru a introduce un text CTRL/A, se va tasta secvența \ CTRL/A. Pentru a introduce în text se va tasta \\ . Pe parcursul introducerii inițiale a unui text, poziția cursorului trebuie să fie aceeași cu cea a Indicatorului. Pe lîngă introducerea inițială a unui text, în mod ecran editorul mai are următoarele funcții :

— Înlocuirea caracter cu caracter a unui text ;

— Inserarea de noi caractere în interiorul unui text ;

— Ștergerea unor caractere din text ;

— Afișarea unor porțiuni din text ce nu încap pe ecran.

Inlocuirea

Dacă cursorul este poziționat pe un caracter din text, introducerea unui caracter de la tastatură va înlocui caracterul de pe ecran și va deplasa cursorul în poziția următoare. Acest lucru continuă pînă ce cursorul ajunge la sfîrșitul liniei. În acest moment, linia este expandată și terminatorul de linie (↑) este deplasat la dreapta. Înlocuirea se face caracter cu caracter exact ca și la scrierea peste un text. Un caracter poate fi înlocuit cu un caracter de control utilizînd funcția de literalizare (\).

Inserarea

În interiorul unui text se poate insera printr-o comandă un caracter sau printr-o altă comandă se poate insera un șir de caractere de lungime oarecare. După inserare, noul text apare imediat în stînga cursorului iar cursorul este poziționat pe același caracter ca și înainte de introducerea comenzii.

Comanda CTRL/A: permite inserarea unui text în interiorul textului editat.

Formatul comenzii este :

CTRL/A *text* **CTRL/A** unde :

CTRL/A este comanda de inserare ;

text reprezintă orice șir de caractere introduse de la tastatură, ca în modul de introducere inițială a unui text.

După deplasarea cursorului în poziția unde se dorește inserarea, se introduce CTRL/A și ecranul este șters începînd cu această poziție permițînd afișarea textului introdus. Dacă se ajunge la ultimul rînd al ecranului, textul se deplasează în sus rînd cu rînd. După terminarea inserării, se introduce CTRL/A și textul ce urma de la poziția cursorului înainte de comandă, este afișat în continuare.

Comanda CTRL/C: permite inserarea unui singur caracter.

Formatul comenzii este :

CTRL/C unde :

CTRL/C este comanda de inserare caracter ;

x este caracterul ce va fi inserat imediat în stînga poziției cursorului. Se poate utiliza funcția de literalizare (\).

Ștergerea

Ca și pentru inserare există două comenzi de ștergere, pentru un număr oarecare de caractere și respectiv pentru un singur caracter. După execuția unei comenzi de ștergere textul este compactat și afișat pe ecran imediat.

Comanda CTRL/Z: permite ștergerea unui număr oarecare de caractere.

Formatul comenzii este :

CTRL/Z *deplasări ale cursorului* **CTRL/Z** unde :

CTRL/Z este comanda de ștergere ;

deplasări ale cursorului reprezintă deplasarea cursorului pînă în poziția ultimului caracter ce se dorește a fi șters. Pentru a șterge o porțiune de text se deplasează cursorul pe poziția primului caracter din această porțiune și se introduce de la tastatură CTRL/Z. Caracterul indicat de cursor este înlocuit cu @. Se deplasează apoi cursorul pînă la ultimul caracter și se introduce din nou CTRL/Z.

Textul marcat de cele două CTRL/Z este șters, textul rămas este compactat și afișat pe ecran. Între cele două caractere de comandă, CTRL/Z, deplasarea cursorului este marcată de înlocuirea temporară a caracterului tipăribil cu (@).

Comanda CTRL/D : permite ștergerea unui singur caracter.

Formatul comenzii este :

CTRL/D

Caracterul indicat de cursor este șters, textul compactat și afișat după modificare.

Afișarea

În mod ecran există trei comenzi de afișare care permit vizualizarea paginii curente față de poziția Indicatorului, a paginii precedente și respectiv a paginii următoare.

O pagină conține 20 linii de text.

Comanda CTRL/V : are ca efect afișarea unei pagini de text relativ la poziția cursorului astfel că acesta se va găsi, după execuția comenzii, în linia a treia. În cadrul liniei, cursorul rămîne pe aceeași poziție.

Formatul comenzii este :

CTRL/V.

Dacă în mod comandă se introduce CTRL/V ca primul caracter dintr-o linie de comandă, CREDIT trece în mod ecran, șterge ecranul și afișează o pagină de text cu cursorul poziționat în linia a treia.

Comanda CTRL/N : are ca efect afișarea paginii următoare, începînd cu ultimele două linii din pagina curentă. Cursorul este poziționat pe primul caracter din linia a treia sau din ultima linie dacă există mai puțin de trei linii.

Formatul comenzii este :

CTRL/N.

Comanda CTRL/P : are ca efect afișarea unei pagini formată din 18 linii ce preced pagina curentă urmate de primele două linii din pagina curentă. Cursorul se poziționează pe primul caracter din linia a treia.

Formatul comenzii este :

CTRL/P.

7.3.4. Editarea în mod comandă

Pentru introducerea unui text cu editorul și pentru efectuarea unor modificări relativ simple, facilitățile disponibile în mod ecran sînt satisfăcătoare în cele mai multe cazuri. Există însă situații în care sînt necesare comenzi „mai puternice" pentru efectuarea unor operații de editare de o mai mare complexitate. În mod comandă CREDIT oferă astfel de facilități. În acest mod de lucru nu se utilizează cursorul ca indicator de text și nici ecranul pentru a vizualiza imediat modificările efectuate. CREDIT este utilizat în acest mod ca și ETX18 sau EDIT avînd însă o serie de facilități suplimentare : operații cu blocuri de caractere, cu fișiere, posibilități de utilizare a macrocomenzilor etc.

Trecerea din mod ecran în mod comandă se face prin comanda Home sau SHIFT/CONTROL/M (pentru tastatură fără litere mici) sau prin introducerea codului 1DH de la tastatură. La intrarea în mod comandă CREDIT afişează (*) pentru a anunţa utilizatorul că a intrat în regim de aşteptare comenzi. Comenzile sînt introduse în Zona Comenzi (primele 4 sau 5 linii de pe ecran) şi dacă se depăşeşte această zonă, comenzile vor fi afişate în continuare pe tot ecranul, Zona text fiind ştearsă în acest scop.

CREDIT permite introducerea unui şir de comenzi separate prin (;). Un şir de comenzi trebuie terminat prin <cr> pentru a fi executat. Ex :

* *comandă ; comandă ; comandă* <cr>

Dacă se doreşte introducerea unui şir de comenzi ce depăşeşte lungimea unei linii, înainte de <cr> se introduce caracterul (&) şi linia poate fi continuată. Linia de continuare începe cu două caractere (*). Ex. :

* *comandă ; comandă ; comandă ;* &<cr>

***comandă ; comandă* <cr>

Editarea unei linii de comenzi se poate face prin ştergerea (eventual repetată) a ultimului caracter prin RUBOUT (sau DEL).

Sintaxa comenzilor este :

* *nume comandă* [*argument*]

Argumentul este opţional şi poate consta din 1, 2 sau 3 parametri ceruţi de comandă. Unele comenzi necesită ca parametri şiruri de caractere. În acest caz un şir de caractere ce este parametru trebuie delimitat pentru a putea fi separat de comandă. CREDIT interpretează primul caracter diferit de spaţiu ca delimitator de început şi următorul caracter identic cu acesta ca delimitator de sfîrşit de şir. Orice caracter ASCII poate fi utilizat ca delimitator exceptînd cr, lf, spaţiu, \ sau escape.

În mod comandă CREDIT utilizează indicatorul în Zona Text dar, spre deosebire de lucrul în mod ecran, poziţia acestuia nu este arătată în mod explicit pe ecran.

Comenzi de poziţionare a indicatorului

Pentru a localiza anumite porţiuni din text, există comenzi de deplasare a Indicatorului înainte sau înapoi faţă de poziţia curentă, cu un număr oarecare de caractere sau linii, sau la un marcaj (tag) definit anterior sau un marcaj permanent.

Comanda L: deplasează indicatorul peste un număr de linii şi-l poziţionează la începutul liniei destinaţie.

Formatul comenzii este :

L [*număr*] unde :

L este numele comenzii ;

număr specifică numărul de linii cu care se deplasează indicatorul. Dacă este pozitiv, indicatorul se deplasează spre sfîrşitul fişierului iar dacă este negativ, spre început. Dacă numărul este 0, indicatorul se poziţionează pe primul caracter din linia curentă iar dacă este omis, pe primul caracter din linia următoare.

Comanda J : deplasează indicatorul peste un număr de caractere sau la poziţia marcajului specificat.

Formatul comenzii este :

J [*număr* | *marcaj*] unde :

J este numele comenzii ;

număr specifică numărul de caractere peste care va fi deplasat Indicatorul şi poate fi pozitiv sau negativ.
marcaj specifică, dacă lipseşte număr, poziţia unui marcaj definit unde va fi deplasat indicatorul.
Dacă *număr* şi *marcaj* sînt omise se considera 1 ca parametru şi indicatorul se deplasează înainte (spre sfîrşitul fişierului) cu o poziţie.

Pentru deplasarea cursorului la poziţia unui marcaj, acesta trebuie definit în prealabil. Există 4 marcaje permanente ce nu pot fi şterse de utilizator :

TT — Începutul fişierului în curs de editare ;
TE — Sfîrşitul acestuia ;
TB — Începutul zonei curente de text din memorie ;
TZ — Sfîrşitul acesteia.

Pe lîngă acestea utilizatorul poate defini sau şterge pînă la 10 marcaje suplimentare. Pentru a redefini un marcaj acesta trebuie şters în prealabil.

Comanda TS : are ca efect definirea unui marcaj la locaţia curentă a Indicatorului. Marcajul este asociat caracterului arătat de Indicator şi deplasarea sau ştergerea caracterului respectiv este însoţită de deplasarea, sau ştergerea marcajului. Dacă există deja un marcaj definit cu acelaşi nume, se afişează un mesaj de eroare.
Formatul comenzii este :
TS*n*
unde :
TS este numele comenzii ;
n este numărul (între 0 şi 9) al marcajului.

Comanda TD : are ca efect ştergerea unui marcaj definit anterior (cu excepţia marcajelor permanente).
Formatul comenzii este :
TD*n*
unde :
TD este numele comenzii ;
n este numărul (între 0 şi 9) al marcajului ce trebuie şters.

Comenzi de afişare şi modificare a textului

Aceste comenzi oferă posibilitatea de prelucrare (editare) efectivă a textului printr-un set de funcţii ca :

— Tipărirea unui număr de linii ;
— Inserarea unei porţiuni de text ;
— Ştergerea unei porţiuni de text ;
— Deplasarea unei porţiuni de text în altă poziţie ;
— Copierea unei porţiuni de text într-o altă poziţie a fişierului editat.

Comanda P : permite tipărirea la consolă a unui număr de linii.
Formatul comenzii este :
P*[n| marcaj]*
unde :
P este numele comenzii ;
n un număr pozitiv sau negativ ce specifică numărul de linii, după sau înainte de Indicator, ce vor fi afişate ;

n=0 se afişează linia curentă pînă la Indicator ;
dacă
n=1 se afişează linia curentă de la Indicator pînă la <cr>.
Dacă se specifică un marcaj (şi nu se specifică număr) prin T urmat de numărul marcajului, se tipăreşte la consolă textul începînd cu poziţia curentă a Indicatorului pînă la marcajul specificat sau, dacă acesta nu este găsit, pînă la TE. Dacă nu se specifică nici un parametru se afişează întreaga linie curentă (ce conţine Indicatorul).

Comanda I : permite inserarea unui şir de caractere oriunde în text, imediat înainte de poziţia Indicatorului, aceasta rămînînd neschimbată.
Formatul comenzii este :
I/*text*/
unde :
I este numele comenzii ;
/ reprezintă delimitatorul de la început şi sfîrşit de text. Acesta poate fi orice caracter diferit de spaţiu, cr, lf, escape, \.
text reprezintă orice şir de caractere cuprins între delimitatori (de obicei /). Dacă delimitatorul de sfîrşit de şir este omis, CREDIT continuă aşteptarea lui pînă la depăşirea zonei de comenzi (2000 de caractere). Orice caracter cuprins între delimitatori este inserat în text.

Comanda DL : permite ştergerea unui număr de linii din text.
Formatul comenzii este :
DL [*număr*]
unde :
DL este numele comenzii ;
număr specifică numărul de linii ce vor fi şterse.
Dacă *număr* este negativ se şterg liniile ce preced poziţia Indicatorului. Dacă este pozitiv se şterg liniile începînd cu poziţia Indicatorului (inclusiv).
Dacă *număr*=0 se şterge linia curentă pînă la poziţia Indicatorului. Dacă *număr* este omis se şterge întreaga linie ce conţine Indicatorul.

Comanda DC : permite ştergerea unui număr de caractere din text.
Formatul comenzii este :
DC [*număr* | *marcaj*]
unde :
DC este numele comenzii ;
număr poate fi pozitiv sau negativ şi specifică numărul de caractere ce vor fi şterse începînd cu poziţia Indicatorului (inclusiv) sau pînă la acesta (exclusiv).
marcaj specifică (dacă *număr* lipseşte) sfîrşitul blocului de caractere ce vor fi şterse. Dacă nu se specifică *număr* sau *marcaj* se şterge caracterul arătat de Indicator.

Comanda XM : mută întreg textul marcat prin două marcaje sau prin număr de linii la poziţia Indicatorului şi şterge textul original din vechea poziţie. Poziţia Indicatorului poate să fie oriunde în cadrul unei linii.

Formatul comenzii este :

XM *marcaj*, { *marcaj* | *număr* }

unde :

XM este numele comenzii ;

Primul marcaj specifică începutul blocului ce trebuie mutat ; Al doilea marcaj trebuie să fie după primul și specifică sfîrșitul blocului marcat. În loc de cel de al doilea marcaj se poate specifica un număr pozitiv sau negativ reprezentînd liniile, după sau înaintea primului marcaj, ce urmează să fie mutate la poziția curentă a Indicatorului. Indicatorul rămîne poziționat pe primul caracter după blocul mutat.

Comanda XC : copiază întreg textul marcat prin două marcaje sau prin număr de linii la poziția curentă a Indicatorului fără a șterge textul original din vechea poziție.

Formatul comenzii este :

XC *marcaj*, {*marcaj* | *număr*}

unde :

XC este numele comenzii ;

Parametrii au aceeași semnificație ca în cazul comenzii XM.

Pentru a facilita localizarea prin context a Indicatorului în fișierul editat CREDIT este prevăzut cu o comandă de căutare a unui șir de caractere. Indicatorul va fi poziționat pe primul caracter după șirul specificat dacă este găsit, dacă nu, rămîne pe poziția de unde a început căutarea. Printr-o altă comandă se poate specifica găsirea unui șir de caractere și înlocuirea acestuia cu un alt șir, Indicatorul fiind poziționat pe primul caracter după acesta.

Pentru ambele comenzi căutarea se poate face în tot fișierul începînd cu poziția Indicatorului sau numai într-o zonă delimitată de parametrii specificați în comandă.

Pentru căutare, un text poate fi specificat exact prin șirul de caractere din care este format, dar există și posibilitatea de specificare pe anumite poziții ale șirului a unor *caractere speciale* care permit identificarea tuturor șirurilor care aparțin la mulțimea determinată de aceste semne speciale. Astfel :

? se identifică cu orice caracter ASCII.

Ex. : ABC?E se identifică cu ABCDE,ABC5E,ABC*E sau orice șir care are în pozițiile 1, 2, 3, 5 caracterele ABCE indiferent de caracterul din poziția 4 ;

CTRL/Y identifică caracterul ce urmează după CTRL/Y cu oricîte caractere identice după CTRL/Y.

Ex. : CTRL/Y0AH se identifică cu 0AH, 00AH, 000AH ;

CTRL/W identifică aceleași caractere indiferent că sînt cu sau fără SHIFT (mari sau mici).

Ex. : CTRL/W mhZ CTRL/W se identifică cu MhZ, mHZ, MHZ, MHZ.

Comanda F : caută un șir de caractere marcat de delimitatori de început și sfîrșit de șir.

Formatul comenzii este :

F *(șir de caractere)* [*marcaj*/*număr*] unde :

F este numele comenzii ;

(șir de caractere) specifică șirul căutat, între delimitatori ; *mar-*

caj/număr reprezintă, împreună cu poziția curentă a Indicatorului, porțiunea de text în care se face căutarea : de la Indicator pînă la marcaj sau numărul de linii după (număr pozitiv), sau înainte (număr negativ) de poziția curentă a Indicatorului.
Dacă număr =0 se caută de la începutul liniei pînă la Indicator.

Comanda S | SQ : caută un șir de caractere și-l înlocuiește cu alt șir.
Formatul comenzii este :
{**S** | **SQ**} *(text vechi/text nou)* [*marcaj/număr*]
unde : S este numele comenzii de înlocuire a textului vechi cu cel nou ;
SQ este numele comenzii ce caută textul vechi, întreabă operatorul și face înlocuirea numai dacă răspunsul este Y sau y. Textul vechi trebuie să fie de lungime diferită de 0 iar textul nou poate fi de lungime 0. Ambele texte trebuie întotdeauna încadrate de delimitatori. Dacă textul vechi nu este găsit se afișează un mesaj de eroare **NOT FOUND ;**
Prin număr sau marcaj se poate fixa Zona din text unde operează comanda. Șirul de caractere reprezentînd textul vechi poate conține caracterele speciale ?, CTRL/Y sau CTRL/W. Execuția comenzii poate fi întreruptă prin introducerea de la tastatură a caracterului ESC.

7.3.5. Tehnici de editare avansate

Facilitățile editorului prezentate în paragrafele precedente acoperă în cea mai mare parte cerințele utilizatorilor. Pe lîngă acestea, CREDIT este prevăzut cu funcții ce permit utilizarea unor tehnici avansate de editare de către utilizatorii cu experiență în editarea cu CREDIT. Aceste funcții se referă la posibilitățile de utilizare a unor macrocomenzi, executarea iterativă a unui șir de comenzi, fișiere de comenzi memorate pe disc, execuția condițională a unor șiruri de comenzi, fișiere de date pe disc, fișiere indirecte de comenzi, comenzi pentru adaptarea contextului de editare la caracteristicile terminalului utilizat.

Macrocomenzi

Macrocomenzile sînt șiruri de comenzi recunoscute de CREDIT care sînt definite și denumite de către utilizator ca macrocomenzi și pot fi apelate sub numele dat la definire. În acest fel se pot grupa șiruri de comenzi care se repetă în cursul editării facilitînd astfel introducerea fără erori a șirurilor lungi de comenzi. La utilizarea în mod comandă se pot defini macrocomenzi cu parametri formali urmînd ca în momentul apelării să se transmită parametrii efectivi. La definirea unei macrocomenzi se permit și apeluri de macrocomenzi, ceea ce permite apelarea recursivă a macrocomenzilor. Utilizatorul trebuie să examineze cu atenție această posibilitate. De asemenea în corpul de definiție pot fi introduse comenzi care schimbă modul de editare (ecran sau comandă), dar trebuie avut în vedere ca terminarea execuției unei macrocomenzi să se facă în același mod în care a început ; astfel se evită execuția într-un mod inadecvat a comenzilor în continuare.

Funcțiile ce permit manipularea macrocomenzilor sînt :
— Definirea unei noi macrocomenzi (MS) ;
— Execuția unei macrocomenzi (MF) ;
— Ștergerea unei macrocomenzi existente (MD) ;
— Afișarea definiției macrocomenzilor existente (?M).

Comanda MS : definește un șir de comenzi ca o macrocomandă cu numele specificat. Dacă acest nume există deja, se emite un mesaj de eroare și definirea nu se efectuează.

Formatul comenzii este :

MS *nume / șir de comenzi /* unde :

MS este numele comenzii ;

nume este un singur caracter ASCII reprezentînd numele macrocomenzii. Acest caracter poate fi orice caracter în afară de cr, lf, escape, \, * și spațiu. Nu este permisă multidefinirea ;

Șir comenzi reprezintă comenzile ce se vor executa la apelarea macrocomenzii. Acesta trebuie delimitat cu un delimitator de început și sfîrșit de șir.

Fiecare parametru formal (dacă este cazul) este specificat prin caracterul (%). La apelare, pentru fiecare (%) trebuie furnizat un parametru efectiv. Aceștia sînt specificați în paranteze și separați prin virgule. Corespondența între parametrii formali și cei efectivi trebuie să fie 1 la 1, altfel se emite un mesaj de eroare. Semnul % poate fi utilizat în textul de definite ca și caracter obișnuit prin dublă literalizare (\\%).

Comanda MF : execută macrocomanda specificată dacă este definită ; dacă nu, se emite un mesaj eroare.

Parametrii efectivi (dacă este cazul) sînt transmiși în ordinea de la definire, delimitați prin paranteze și separați de virgule.

Formatul comenzii este :

MF *nume* [(*param* [,. . ., *param*])]

unde :

MF este numele comenzii ;

param... reprezintă parametrii efectivi (opțional) ce sînt transmiși macrocomenzii. Un parametru nul se poate transmite prin lipsa textului între virgulele de separare.

Virgula poate fi introdusă în text ca parametru prin literalizare (\,).

Comanda CTRL/F : expandează o macrocomandă exact ca și MF dar trebuie notat că CTRL/F se execută în mod ecran și deci comenzile din șirul de definire trebuie să fie alese corespunzător modului ecran. De asemenea trebuie notat că nu se pot transmite parametri efectivi.

Formatul comenzii este :

CTRL/F *nume*.

Comanda MD : șterge toate macrocomenzile definite sau numai pe cea specificată prin nume.

Formatul comenzii este :

MD {*nume* | *} unde :

MD este numele comenzii ;

nume specifică numele macrocomenzii ce se va șterge ;

* reprezintă indicația că trebuie șterse toate macrocomenzile.

Comanda ?M : afișează numele și definiția tuturor macrocomenzilor.

Formatul comenzii este :

?M.

Execuția iterativă a comenzilor

Un șir de comenzi poate fi executat în mod repetat prin delimitarea acestuia de caracterele (<, >) și specificarea condiției de terminare a execuției.

Formatul este :

{n | ! } <șir comenzi> unde :

n reprezintă numărul maxim de execuții a șirului de comenzi ;

! execuția șirului de comenzi începe cu poziția Indicatorului și se termină cînd se ajunge la sfîrșitul fișierului. Se pot prevedea pînă la cinci nivele de paranteze unghiulare (<, >).

Comanda EL : permite terminarea execuției unei bucle de comenzi specificată prin (<, >). EL trebuie introdusă în șirul de comenzi astfel ca să se execute la îndeplinirea unei condiții, forțînd astfel ieșirea din bucla de comenzi și continuarea execuției comenzilor după buclă. Execuția condițională a comenzilor va fi prezentată în continuare.

Comanda QU : permite atribuirea unei valori logice (adevărat sau fals) unui semafor (Query-flag) ce poate fi testat cu alte comenzi. Astfel la execuția comenzii QU, se afișează pe ecran semnul întrebării (?) și dacă răspunsul de la tastatură este Y sau y semaforul primește valoarea adevărat altfel primește valoarea fals.

Comanda QT : execută comanda următoare dacă semaforul Query este adevărat, altfel sare peste aceasta.

Formatul de utilizare a comenzii este :

QT *; { comandă | <șir comenzi> }* unde :

QT este numele comenzii ;

Comandă | <șir comenzi> reprezintă comanda sau bucla de comenzi ce va fi executată numai dacă semaforul este adevărat, altfel execută comanda următoare lui comandă sau <șir comenzi>.

Comanda QF : execută comanda (sau bucla de comenzi) următoare dacă semaforul Query este fals, altfel execută comanda care urmează după cea imediat următoare lui QF.

Formatul de utilizare este :

QF *; { comandă | <șir comenzi> }*

Pentru execuția condiționată a unor comenzi se poate utiliza ca și condiție de test rezultatul (adevărat sau fals) al execuției comenzilor de căutare a unui șir și de căutare și înlocuire (F, S și SQ). Pentru aceasta CREDIT definește un semafor (Yes-flag) căruia îi atribuie valoarea adevărată sau fals, după cum șirul specificat este găsit sau nu. Acest semafor poate fi testat cu comenzile YT și YF.

Comanda YT : execută comanda (sau bucla de comenzi) următoare numai dacă semaforul Yes este adevărat, altfel sare la comanda următoare celei imediat următoare lui YT.

Formatul de utilizare este :

YT *; { comandă | <șir comenzi> }.*

Comanda YF : execută comanda (sau bucla de comenzi) următoare **numai** dacă semaforul Yes are valoarea logică fals.

Formatul de utilizare este :

YF *;* { *comandă* | *<şir comenzi>* }.

Comanda U : permite afişarea unui mesaj la terminal.

Formatul comenzii este :

U /**text**/ unde :

U este numele comenzii ;

/**text**/ reprezintă mesajul ce va fi afişat la comandă. Textul trebuie delimitat cu acelaşi caracter la început şi sfîrşit.

Utilizarea fişierelor de comenzi

Un şir de comenzi, inclusiv definiri şi apeluri de macrocomenzi, poate fi memorat sub forma unui fişier SFDX şi executat exact ca şi cum ar fi introdus de la tastatură. Macrocomenzile definite într-un fişier pot fi utilizate în continuare pe tot cursul editării. Numele fişierului poate fi CREDIT.MAC şi atunci nu mai trebuie specificat în comanda de lansare în execuţie (CREDIT PROG.ASM MACRO), încărcarea lui se face implicit de pe discul care conţine programul CREDIT sau poate avea un alt nume specificat explicit în comanda de lansare în execuţie (CREDIT PROG.ASM MACRO (: F3 : COM.MAC)).

Un fişier de comenzi poate fi încărcat şi cu o comandă specială.

Comanda G : încarcă un fişier de comenzi.

Formatul comenzii este :

G *nume de fişier ;*

Fişierul de comenzi poate la rîndul lui să conţină comanda G. Procedeul poate fi utilizat pe oricîte nivele deoarece un singur fişier este deschis la un moment dat.

Utilizarea fişierelor de date

Pentru a facilita editarea unor fişiere foarte mari, CREDIT permite utilizarea unor fişiere de date auxiliare care pot fi citite/scrise în timpul unei editări. La un moment dat pot fi deschise cel mult două fişiere, unul pentru citire şi celălalt pentru scriere. În continuare se prezintă comenzile care permit utilizarea fişierelor de date.

Comanda OR : deschide un fişier pentru citire. Fişierul specificat trebuie să nu fie deja deschis (exceptînd :CI : care este deschis pentru citire permanent).

Indicatorul se poziţionează la începutul fişierului.

Formatul comenzii este :

OR *nume fişier* unde :

OR este numele comenzii ;

nume fişier specifică fişierul ce trebuie deschis pentru citire.

Comanda OW : deschide un fişier pentru scriere. Fişierul trebuie să nu fie deja deschis, exceptînd :CO : care e deschis în permanenţă.

Formatul comenzii este :

OW *nume fişier.*

Dacă nume fişier există deja, acesta va fi şters. Indicatorul se poziţionează la începutul fişierului.

Comanda B : deplasează Indicatorul extern la începutul fişierului deschis pentru citire. Dacă nu există un astfel de fişier, comanda nu are efect.

Formatul comenzii este :

B

Comanda R : Citeşte un număr de linii din fişierul extern deschis pentru citire începînd cu poziţia Indicatorului extern şi inserează textul citit în textul în curs de editare începînd cu poziţia Indicatorului. Textul citit nu este şters iar Indicatorul extern este poziţionat pe primul caracter ce urmează blocului citit. Indicatorul intern se poziţionează pe primul caracter ce urmează blocului inserat.

Formatul comenzii este :

R [*n*] unde :

R este numele comenzii ;

n specifică (opţional) numărul de linii citite. Dacă n lipseşte se citeşte o linie. Dacă n=0 comanda nu are efect.

Comanda W : scrie un număr de linii într-un fişier extern deschis pentru scriere.

Formatul comenzii este :

W [*număr*] unde :

W este numele comenzii ;

număr reprezintă numărul de linii ce vor fi scrise.

Dacă număr este negativ se scriu liniile ce preced poziţia Indicatorului iar dacă este pozitiv, liniile ce urmează. Se scrie inclusiv linia care conţine Indicatorul în ambele cazuri. Dacă n=0 se scrie linia curentă pînă la poziţia Indicatorului. Comanda W nu modifică poziţia Indicatorului intern.

Comanda CR : închide fişierul deschis pentru citire.

Formatul comenzii este :

CR

Comanda CW : închide fişierul deschis pentru scriere.

Formatul comenzii este :

CW

Dacă nu există fişier deschis, CR sau CW nu au efect.

Pentru a putea adapta contextul în care se execută CREDIT la caracteristicile terminalului s-a prevăzut o comandă care poate fi executată imediat după lansarea în execuţie. Astfel se pot modifica :

— numărul de linii ce pot fi afişate pe ecran ;
— codurile de intrare şi ieşire pentru deplasarea cursorului ;
— codurile de intrare şi ieşire pentru funcţiile de ştergere a ecranului ;
— caracterul afişat ca terminator de linie ;
— coloanele pentru tabulare orizontală ;
— codurile utilizate pentru caracterele *„generice“* (?, CTRL/W, CTRL/Y) ;
— suprimarea mesajelor de eroare la execuţia comenzilor F şi S, SQ.

Comanda A : modifică contextul de execuţie a editorului.

Formatul comenzii este :

{**AF** | **A** }*cod*=*valoare* unde :

A | AF este numele comenzii ;

cod este format din 1 sau 2 caractere reprezentînd funcția ce urmează să fie modificată ;

valoare reprezintă valoarea corespunzătoare ce se atribuie funcției selectate pentru modificare. Spațiile după (=) sînt considerate ca atare (deci nu se ignoră).

Valorile posibile pentru *cod* sînt :

AL=terminatorul de linie ce se va afișa la fiecare cr, lf (un singur caracter) ;

AV=numărul de linii reprezentînd dimensiunea ecranului (22, 23, 24, 25) ;

AS=afișarea (T) sau suprimarea (F) mesajelor de eroare emise la execuția comenzilor F, S, SQ ;

AT=poziția coloanelor pentru tabulare (0—79). Implicit se consideră 8 ;

AFWA=codul utilizat pentru identificarea cu orice număr de caracter. Implicit este CTRL/Y ;

AFWC=codul utilizat pentru identificarea caracterelor cu sau fără SHIFT. Implicit este CTRL/W ;

AFWI=codul utilizat pentru identificare cu orice caracter. Implicit este (?) ;

AFBK=codul pentru suprimarea afișării unui caracter pe ecran. Implicit este spațiu (blanc=20H).

Pentru următoarele funcții *valoare* este formată din două caractere. Dacă cel de al doilea caracter nu este introdus, se consideră 0. Orice caracter poate fi literalizat pentru a putea fi specificat ca valoare. Dacă editarea fișierului de comenzi se face în mod ecran, escape nu poate fi literalizat.

AFCU=codul generat de tastatură pentru deplasarea cursorului în sus ;

AFCD=codul generat de tastatură pentru deplasarea cursorului în jos ;

AFCL=codul generat de tastatură pentru deplasarea cursorului la stînga ;

AFCR=codul generat de tastatură pentru deplasarea cursorului la dreapta ;

AFMU=codul (1 sau 2 caractere) ce trebuie trimis în ecou pentru deplasarea cursorului în sus ;

AFMD=codul ce trebuie trimis în ecou pentru deplasarea cursorului în jos ;

AFML=codul ce trebuie trimis în ecou pentru deplasarea cursorului la stînga ;

AFMR=codul ce trebuie trimis în ecou pentru deplasarea cursorului la dreapta ;

AFES=codul de ștergere a întregului ecran ;

AFER=codul de ștergere a restului ecranului ;

AFEL=codul de ștergere a restului liniei.

Comanda ?A: afișează codurile curente pentru funcțiile modificabile (prezentate mai sus).

Utilizarea editorului cu SUBMIT

O sesiune de editare completă, sau alte operații implicînd funcțiile sistemului de operare inclusiv editarea, pot fi memorate într-un fișier pe disc și executate cu SUBMIT :

SUBMIT *nume fișier*.**CSD.**

Trebuie avute în vedere următoarele particularități :

— CREDIT se execută numai în mod comandă ;
— EQ nu necesită Y sau y de la tastatură ;
— QU și SQ emit mesajul de întrebare direct la consolă ;
— Cel mult două fișiere pot fi deschise la un moment dat ;
— CREDIT nu recunoaște comanda CTRL/V ;
— Caracterele CTRL/E și CTRL/P sînt utilizate de SUBMIT.

7.3.6. Mesajele de eroare emise de editor

În general erorile sînt specifice modului de lucru-ecran sau comandă cu excepția următoarelor trei erori ce nu depind de modul de lucru.

— **INSUFICIENT MEMORY** apare în momentul în care memoria internă disponibilă devine mai mică de 64 octeți ;
— **CREDIT ERROR** apare la detectarea de către CREDIT a unei erori interne ;
— **WARNING : DISK FULL** este un mesaj de avertizare că mai există prea puțin spațiu disponibil pe disc și editarea poate continua cu riscul ca anumite comenzi să nu poată fi executate și să se compromită editarea.

Erorile în mod ecran

În acest mod de lucru pot apare ca mesaje explicite de eroare doar cele legate de utilizarea macrocomenzilor. Erorile de tip comenzi ilegale sînt semnalate prin avertizorul sonor al terminalului și pot apare în următoarele situații :

— Încercarea de deplasare a cursorului în afara limitelor ecranului ;
— Încercarea de efectuare a unei modificări de tip ,ștergere într-o zonă fără text și cu poziția cursorului diferită de cea a Indicatorului ;
— Încercarea de introducere în cadrul comenzilor CTRL/A sau CTRL/C a unui caracter netipăribil fără să fie literalizat ;
— Utilizarea funcției RUBOUT(DEL) în afara comenzii CTRL/A ;
— Introducerea celui de al doilea marcaj de ștergere text, CTRL/Z înaintea primului ;
— Introducerea unui cod diferit de CTRL/Z sau codurile de deplasare a cursorului în cadrul comenzii CTRL/Z ;
— Încercarea de ștergere sau modificare a caracterului reprezentînd terminatorul de fișier.

În mod ecran pot apare erori sintactice la referirea unor macrocomenzi nedefinite sau cu definiri eronate.

Erorile în mod comandă

O eroare detectată la lansarea în execuție, la editarea în mod comandă determină afișarea unui mesaj de eroare și a liniei eronate pînă la punctul unde s-a detectat eroarea.

Erorile posibile semnalate de CREDIT sînt următoarele :

UNCLOSED STRING

Delimitatorul de sfîrșit de șir nu a fost găsit pînă la sfîrșitul liniei de comandă.

FILE ACCES ERROR

Încercarea de a face acces la un fișier în mod inadecvat. (Scriere la un fișier protejat, citire de la un fișier de ieșire etc.).

FILE IN USE

Fișierul este deja deschis deci nu mai poate fi deschis pînă nu este închis mai înainte.

UNRECOGNIZED COMMAND

Introducerea unei comenzi nerecunoscută de CREDIT.

IMPROPER OPERAND

Argumentul comenzii este eronat.

MISSING OPERAND

Comanda nu conține operandul cerut de formatul acesteia.

ILLEGAL VALUE

Un operand de tip număr conține caractere nenumerice sau este înafara intervalului —32767 : +32767

ITTERATION ERROR

Paranteze unghiulare incorect împerecheate sau specificarea numărului de iterații este greșită.

ARGUMENT MISMATCH

Nepotrivirea numărului de argumente efective și formale în timpul espandării unei macrocomenzi.

BUFFER FULL

Încercarea de a introduce peste 2 000 de caractere în zona de comenzi. Poate să apară la I, G, MF, la introducerea unei linii de comenzi.

TERMINATOR EXPECTED

O comandă a fost terminată prin altceva decît (;), (>) sau CR, LF.

ILLEGAL NAME

Nume de marcaj, fișier sau macrocomandă incorect.

DOESN'T EXIST

Numele de marcaj, fișier sau macrocomandă specificat nu există.

ALREADY EXISTS

Numele fișierului, marcajului sau macrocomenzii există deja.

TAG POSITION

Marcajul de sfîrșit pentru comenzile DC, XC sau XM este înaintea marcajului de început.

WARNING : DISK FULL

CREDIT apreciază că nu prea mai există spațiu pe disc suficient pentru a continua editarea. Se poate continua totuși, dar pe răspunderea utilizatorului.

7.3.7. Fişiere utilizate de CREDIT

Pe lîngă fişierele de comenzi şi de date controlate de utilizator există mai multe fişiere temporare utilizate de CREDIT ca memorie tampon pentru executarea unor comenzi. Trebuie avut în vedere faptul că SFDX permite deschiderea a cel mult 6 fişiere la un moment dat.

Astfel, dacă se lansează CREDIT pentru editarea unui fişier existent, vor fi deschise trei fişiere :

— Fişierul specificat în comanda de lansare ;
— Un fişier de ieşire CREDT1.TMP care conţine rezultatul editării pînă ce se execută EX. In acest moment CREDT1.TMP este redenumit cu numele fişierului destinaţie specificat (implicit sau explicit) la lansare ;
— Un fişier de ieşire CREDT2.TMP utilizat doar la execuţia unor comenzi de deplasare a Indicatorului spre începutul fişierului într-o zonă scrisă deja pe disc.

Unele comenzi utilizează explicit fişiere de disc .Ex. :

— G utilizează un fişier de comenzi specificat de utilizator ;
— XC, XM utilizează un fişier temporar CREDT3.TMP ;
— OR, OW utilizează fişiere specificate de utilizator ;
— R, W utilizează fişierele deschise cu OR şi OW.

Într-o utilizare normală a editorului nu se depăşeşte limita de 6 fişiere deschise simultan, limită impusă de zona de memorie rezervată pentru zonele tampon.

La utilizarea editorului cu SUBMIT există posibilitatea de a avea deschise fişierul indirect (SUBMIT), fişierul ce se editează, două fişiere .TMP, două fişiere deschise cu OR şi OW şi la execuţia unei comenzi ce necesită CREDT3.TMP se detectează o eroare fatală care reiniţializează SFDX.

Cele trei fişiere CREDT ?.TMP au nume rezervate şi nu pot fi utilizate în alt scop. La terminarea corectă a unei sesiuni de editare (prin EX sau EQ) aceste fişiere sînt şterse sau redenumite. Fişierele temporare se deschid pe aceeaşi unitate pe care va fi scris fişierul destinaţie, după editare.

Deplasarea indicatorului înapoi într-un fişier foarte mare este o operaţie consumatoare de timp şi spaţiu pe disc. Într-o astfel de situaţie este mai convenabil încheierea editării şi reluarea după aceea. *Pentru editarea unor fişiere foarte mari se recomandă ca fişierul sursă şi fişierul destinaţie să fie pe unităţi separate.* Trebuie notat că fişierele temporare sînt deschise pe discul destinaţie.

7.3.8. Exemplu de utilizare a editorului CREDIT

În acest exemplu se arată în special posibilităţile de lucru cu macrocomenzi ale editorului CREDIT. Fişierul MACRO prezentat conţine toate comenzile pentru crearea şi utilizarea unor biblioteci de programe sursă cu ajutorul editorului CREDIT.

7.3.8 Exemplu

```
MSA?OR%;JTE;I/##%##
/;R5000;CR;I/####
/?
MSL?JTT;AS=T;!<F/##/;YF;EL;LO;P;F/####/>;AS=F?
MSF?JTT;F/##%##/;LO?
MST?MFF(%);YT;MFP?
MSP?TS1;F/####/;LO;J-1;TS2;JT1;PT2;TD2;TD1?
MSW?MFF(%);YT;MFY(%)?
MSY?L:TS1;F/####/;LO;J-1;TS2;JTE;L-1;TS3;L;XCT1,T2;JT3;L;
CW%;W5000;CW;JT3;L;DCTE;JT1;TD1;TD2;TD3?
MSD?MFF(%);YT;MFZ;?
  SZ?TS1;F/####/LO;LO;TS2;JT1;DCT2;DL;TD1;TD2?
U!
SOURCE LIBRARIAN V1.1
DO YOU WANT INSTRUCTIONS(Y/N)!:QU;QT;GSLIB.HLP;AFCR='
```

Exemplul de mai sus permite crearea si intretinerea unor biblioteci de programe sursa

```
U/AVAIABLE MACROS:
A(FILE NAME,MODULE NAME) = ADD A MODULE FROM A SPECIFIED FILE
L = LIST NAMES OF ALL MODULES
T(MODULE NAME) = LIST THE CONTENT OF A SPECIFIED MODULE
W(MODULE NAME,FILE NAME) = WRITE A MODULE INTO A FILE
F(MODULE NAME) = MOVE CURSOR TO THE BEGINIG OF A MODULE
D(MODULE NAME) = REMOVE A MODULE FROM THE LIBRARY

PRESS RETURN TO CONTINUE/;QU
```

Capitolul 8

Utilizarea limbajului de asamblare

La începutul acestui capitol se descrie detaliat asamblorul MAC18 orientat pe lucrul cu benzi (perforate sau magnetice). În continuare se prezintă macroasamblorul ASM80 cu accent pe diferențele față de MAC18.

8.1. Macroasamblor MAC18

8.1.1. Prezentarea generală a macroasamblorului

Macroasamblorul MAC18 este un program care permite utilizatorului asamblarea programelor sale și generarea codului obiect pe un suport extern.

Asamblarea este o funcție programată pentru transformarea unui program sursă scris în limbaj de asamblare în program obiect, în limbaj mașină, direct executabil de către calculator.

Ținînd seama de particularitățile limbajelor de asamblare, față de limbajul mașină această traducere înseamnă de fapt : atribuirea de valori simbolurilor, expresiile simbolice trebuiesc calculate și primesc valori, codurile simbolice ale operațiilor primesc corespondentul în limbaj mașină, adresele simbolice sînt localizate în memorie, macroinstrucțiunile se înlocuiesc cu secvențele de instrucțiuni corespunzătoare, directivele se materializează prin diferite acțiuni pe care le efectuează asamblorul.

Practic acest proces se desfășoară în felul următor : textul sursă stocat pe cartele, bandă perforată sau alt suport (casetă, disc) într-un fișier de lucru, este citit de către asamblor și analizat instrucțiune cu instrucțiune.

Traducerea, în funcție de complexitatea limbajului, se face printr-o singură explorare a textului sursă de către asamblor sau mai multe explorări succesive. Una din caracteristicile importante ale unui asamblor este numărul de citiri ale textului sursă, necesare pentru asamblarea programului sursă. Fiecare citire a programului sursă reprezintă o fază a procesului de asamblare.

Macroasamblorul MAC18 este un asamblor în două faze.

În prima fază asamblorul construiește o tabelă de simboli (TS) în care se trece fiecare simbol folosit de programul asamblat împreună cu valoarea prin care se definește și care i se atribuie simbolului, pentru ca în faza a 2-a să se treacă la traducerea programului sursă în limbaj mașină, folosindu-se în acest scop tabela de simboli construită în prima fază.

Dacă în dorința de a se face un limbaj de asamblare cît mai evoluat se introduc prea multe facilități, reducîndu-se unele din restricțiile impuse, atunci se poate ajunge la situația în care numai 2 faze să nu fie suficiente și să se treacă la asamblorul în 3 sau mai multe faze. Pe de altă parte, este de dorit ca numărul de faze să fie cît mai mic pentru ca procesul de asamblare să fie cît mai rapid. Asamblorul care lucrează în două faze reprezintă o soluție de compromis acceptabilă, între nivelul de evoluție al limbajului și timpul de asamblare.

Totuși asamblarea în 2 faze impune anumite restricții pentru programator. De exemplu, pentru directiva de echivalare a unui simbol EQU, o construcție de forma :

A1 EQU B1+3

B1 EQU 4

este semnalată ca eroare de fază și poate genera alte erori la folosirea lui A1.

De exemplu : **ORG A1** va fi semnalată ca eroare de fază.

Simbolii definiți prin **EQU** trebuie să conțină în expresia de definiție numai simboli ce au fost definiți anterior. Această restricție apare din necesitatea ca la sfîrșitul primei treceri tabela de definire a simbolilor utilizați să fie completată în întregime. Astfel în momentul întîlnirii liniei :

A1 EQU B1+3

B1 nu a fost definit în tabelă, deci este dat ca simbol nedefinit și i se atribuie valoarea zero.

În faza a 2-a simbolul B1 este definit în tabelă și la întîlnirea liniei **'A1 EQU B1+3'** expresia ia valoarea 7 care nu mai coincide cu valoarea 3 din tabelă, semnalîndu-se eroare.

De asemenea este necesară apariția corpului de definire a unei macroinstrucțiuni înainte de apelarea ei pentru a se ști cîți octeți va ocupa expandarea.

Este evident că în scrierea corectă a programelor în limbaj de asamblare, se presupune în primul rînd respectarea riguroasă de către programator a tuturor regulilor de scriere a cuvintelor și a frazelor utilizate în program. Programele scrise în limbaj de asamblare nu sînt scutite totuși de erori de scriere. Erorile care se depistează sînt semnalate programatorului pentru a le putea corecta. Erorile depistate în textul sursă nu trebuie să conducă la abandonarea asamblării. Asamblorul trebuie conceput în așa fel încît să continue tratarea întregului text sursă, pentru a depista cît mai multe erori.

Macroasamblorul MAC18 lucrează în două faze ; faza a doua poate cuprinde mai multe acțiuni în funcție de opțiunea utilizatorului :

— se poate obține listarea textului sursă asamblat ;
— se poate genera programul obiect direct executabil pe un suport extern ;
— se pot combina acțiunile anterioare într-un singur pas.

Faza 1

În procesul de traducere a textului în limbaj mașină asamblorul trebuie să facă o clasificare și o gestiune a tuturor simbolilor întîlniți, care pot fi globali sau locali, definiți sau nedefiniți.

Sarcina principală a primei faze este de a construi o tabelă de simboli cu numele definite în programul sursă și de a asocia fiecărui nume o adresă de memorie sau o valoare.

Macroasamblorul, avînd facilitatea de lucru cu macroinstrucțiuni definite de utilizator, poate lucra în mai multe moduri (stări) distincte :

1. Asamblarea unei linii sursă de pe suport extern ;
2. Definire corp de macroinstrucțiune ;
3. Parametri formali ai unei macroinstrucțiuni ;
4. Culegere corp de macroinstrucțiune ;
5. Sfîrșit definire corp de macroinstrucțiune ;
6. Apel macroinstrucțiune ;
7. Parametri actuali la apelul unei macroinstrucțiuni ;
8. Sfîrșit corp de macroinstrucțiune ;

1. Starea de asamblare obișnuită nu pune probleme dificile în procesul de asamblare. Etichetele se introduc în tabela de simboli utilizați, în momentul întîlnirii lor. Etichetele căutate nu sînt referite, calculul care se face este în legătură cu avansarea contorului programului (PC).

Acesta este incrementat cu 1, 2, 3 pentru instrucțiunile de 1, 2, 3 octeți și este modificat de directivele :

ORG *EXPR*
DS *EXPR*

unde *EXPR* este o expresie aritmetică-logică ce conține numai simboli anterior definiți, sau este avansat cu 2n octeți la întîlnirea directivelor :

DS $data_1, data_2, \ldots\ data_n$

DW $data_1, data_2, \ldots\ data_n$

sau cu n octeți pentru :

DB $'c_1c_2 \ldots c_n'$

2. Starea de definire corp de macroinstrucțiune completează în tabela de simboli a utilizatorului numele macroinstrucțiunii, tipul ei și o adresă de trimitere la sfîrșitul tabelei de simboli unde se va depune corpul de definiție. Se marchează un indicator pentru a permite copierea corpului macroinstrucțiunii în tabela de simboli.

3. Starea de parametri formali impune completarea în tabela de simboli a numelor acestor parametri și a ordinii lor în șirul de definire (primul, al 2-lea etc.).

4. Starea de culegere corp macro presupune citirea liniilor program ce compun corpul macroinstrucțiunii, copierea lor fără nici un fel de analiză în tabela de simboli și substituirea, ori de cîte ori se întîlnește un parametru formal, cu un cod special care să permită, la apeluri de macro, substituirea parametrilor formali cu parametrii actuali.

5. Starea de sfîrșit definire de corp macro presupune copierea corpului macroinstrucțiunii într-o zonă specială a tabelei de simboli și completarea adresei de legătură a numelui macro din tabela de simboli utilizator cu această nouă zonă precum și resetarea indicatorului de copiere corp macro.

6. Starea de apel macroinstrucţiune impune, datorită facilităţilor care permit apel la macroinstrucţiune în alt corp de macro, următoarele acţiuni :

— salvarea indicatorilor de lucru cu macro şi de început şi sfîrşit de definiţie într-o zonă de date de tip stivă ;

— incrementarea nivelului de adîncime al macroinstrucţiunii ;

— iniţializarea numărului de parametri actuali macro ;

— setarea indicatorului de aducere linii program din TS şi nu de pe suport extern.

7. Starea de parametri actuali macro presupune completarea unei tabele cu numele sau valoarea parametrilor de apel a macro instrucţiunii pentru a se permite substituirea, ori de cîte ori se întîlneşte în corpul macro, a unui parametru formal cu valoarea lui reală.

8. Starea de sfîrşit de apel decrementează nivelul de apel de macroinstrucţiune iar dacă acest indicator este 0 se resetează indicatorul de aducere a liniilor programului din TS pentru a se citi mai departe de pe suport extern.

Faza a doua

Sarcina fazei a doua este de a genera cod obiect pe baza tabelelor completate în prima fază.

În starea de asamblare obişnuită se verifică dacă valoarea etichetelor definite în prima fază coincide cu valoarea cu care se întîlnesc etichetele în faza a doua.

Stările de definire corp de macro şi parametri formali sînt ignorate verificîndu-se doar corectitudinea adreselor în corpul de macro.

În rest stările sînt identice.

8.1.2. **Setul de instrucţiuni şi formatul instrucţiunilor**

Sintaxa instrucţiunilor

Formatul programelor sursă este liber dar trebuie să respecte un set minimal de reguli.

O instrucţiune are pînă la 4 părţi distincte numite cîmpuri.

Cîmpul 1 — Cîmpul de etichetă cuprinde un nume utilizat pentru referirea simbolică a instrucţiunilor şi datelor ;

Cîmpul 2 — Cîmpul cod de operaţie ;

Cîmpul 3 — Cîmpul operand (operanzi) conţine adrese sau informaţii despre datele necesare pentru cîmpul de cod ;

Cîmpul 4 — Cîmpul de comentarii. Este prezent numai pentru programator şi este ignorat de asamblor. Programatorul poate folosi cîmpul de comentariu pentru a descrie operaţiile şi a face astfel programul mai uşor de înţeles.

Formatul fiind liber cîmpurile pot fi separate printr-un număr arbitrar de blancuri.

Cîmpul etichetă — este opţional. Dacă este prezent poate fi format din 1—5 caractere alfanumerice, primul caracter fiind literă, '@' sau '?'. După etichetă este obligatorie prezenţa caracterului ':' (Excepţie fac pseudoinstrucţiunile EQU, SET, MACRO). Dacă eticheta conţine mai

multe caractere decît 5, celelalte sînt acceptate dar sînt ignorate pînă la detecția caracterului ':' sau separator (blanc).

Asamblorul nu permite folosirea ca nume de etichete a simbolilor rezervați de asamblor : mnemonice pentru instrucțiuni, nume de pseudoinstrucțiuni, nume de registre etc.

Cîmpul cod de operație — conține un nume care identifică o operație mașină, o pseudoinstrucțiune, o directivă sau un apel de macro.

Cîmpul operand — informația din acest cîmp este utilizată în conjuncție cu cîmpul cod de operație pentru a defini precis operația ce urmează a fi asamblată. În funcție de cîmpul de operație, cîmpul operand poate fi absent sau poate consta din 1 sau 2 operanzi separați prin virgulă.

Sînt 4 tipuri de informații care pot fi recunoscute ca entitate în cîmpul operand iar informația poate fi specificată în 9 moduri.

Tipuri de informație : a) Registre ;
b) Pereche de registre ;
c) Date imediate ;
d) Adrese de memorie.

Specificarea se poate face prin :
1. Date hexazecimale ;
2. Date zecimale ;
3. Date octale ;
4. Date binare ;
5. Contor program ($) ;
6. Constante ASCII ;
7. Etichete asignate ca valoare ;
8. Etichete de instrucțiuni ;
9. Expresii.

1. Date hexazecimale — Fiecare număr hexazecimal trebuie urmat de litera 'H' și trebuie să înceapă cu o cifră (0—9).
2. Date zecimale — Fiecare număr zecimal poate fi urmat opțional de litera 'D'.
3. Date octale — fiecare număr octal (compus din cifrele 0—7) trebuie urmat de una din literele 'O' sau 'Q'.
4. Date binare — Fiecare număr binar trebuie urmat de litera 'B'.
5. Contorul de program — Pentru a se specifica valoarea curentă a contorului de program se folosește caracterul $.
6. Constante ASCII — Șiruri de caractere ASCII incluse în ghilimele (').
7. Etichete asignate ca valoare — Următoarele asignări sînt valabile în asamblor și sînt totdeauna active.

B	asignat cu 0	desemnează registrul B
C	1	C
D	2	D
E	3	E
H	4	H
L	5	L
M	6	referință de memorie
A	7	registrul A
PSW	6	indicatorii de condiții și registrul A
SP	6	registrul SP

Codurile de instrucţiuni sînt asignate cu valoarea lor corespunzătoare tabelei de coduri neasamblate (fără parametri) astfel :

MOV MOV B, B (40H)
JMP JMP ADR (0C3H)
LXI LXI B, *data* (01H)

De exemplu pentru :

DB (MOV B, B), (JMP ADR), 01, 00, 0

va fi generat şirul : 40C3010000.

8. Etichete care apar în cîmpul etichetă al unei instrucţiuni.

9. Expresii aritmetice şi logice cuprinzînd date de tip (1)—(8) legate prin operatori aritmetici +, —, *, /, MOD sau logici NOT, AND, OR, XOR, SHL, SHR precum şi '(',')' pentru a marca priorităţile de evaluare.

Descrierea operatorilor :

MOD — produce restul întreg obţinut prin împărţirea primului operand la al 2-lea ;

NOT — produce complementarea fiecărui bit al operandului ;

AND — produce „ŞI" logic al operanzilor bit cu bit ;

OR — produce „SAU" logic al operanzilor bit cu bit ;

XOR — produce „SAU EXCLUSIV" al operanzilor bit cu bit ;

SHR-SHL — operatori logici de deplasare dreapta sau stînga bit cu bit al primului operand cu un număr de poziţii dat de al 2-lea operand.

Programatorul trebuie să se asigure că rezultatul evaluării expresiilor este o valoare adecvată (să nu depăşească valoarea cerută de context).

De asemenea o instrucţiune în paranteze este o expresie corectă şi valoarea generată este chiar codul instrucţiunii. Operatorul generează expresii care sînt evaluate cu următoarele priorităţi :

1. paranteze (,) (prioritate maximă) ;
2. *, /, MOD, SHL, SHR ;
3. +, — (unari sau binari) ;
4. NOT ;
5. AND ;
6. OR, XOR (prioritate minimă).

În cazul mai multor paranteze sînt evaluate mai întîi expresiile dintre parantezele cu nivelul de adîncime maxim.

Tipuri de informaţie

a) Registru — orice expresie cu valoare finală v, $0 \leq v \leq 7$;
b) Pereche de registre —
   B—B şi C
   D—D şi E
   H—H şi L
   PSW — indicatorii de condiţii şi registrul A
   SP — cei 16 biţi ai indicatorului stivei

Pentru pereche de registre se admite orice valoare binară pară 2n, $0 \leq 2n < 7$;

c) Date imediate — sînt ataşate codului de operaţie în programul obiect. Pentru date imediate pe 1 octet se admite orice expresie aritmetică a cărei valoare v, $0 \leq v \leq 255$, $FF00 \leq v \leq FFFF$;

d) Adresă de 16 biți sau o etichetă a unei alte instrucțiuni.
Cîmpul comentariu trebuie să înceapă întotdeauna cu ';'.

Instrucțiuni recunoscute de asamblor.

Instrucțiunile recunoscute de asamblor sînt în număr de 78 și sînt prezentate în capitolul 2.

8.1.3. Pseudoinstrucțiuni recunoscute de asamblor

Macroasamblorul MAC18 recunoaște, pe lîngă cele 78 de instrucțiuni mașină, 12 pseudoinstrucțiuni care măresc puterea asamblorului.

O pseudoinstrucțiune este scrisă în aceeași manieră ca și o instrucțiune mașină însă nu se generează cod obiect. Scopul pseudoinstrucțiunilor este de a furniza asamblorului informații care să fie folosite la generarea codului obiect.

În general formatul unei pseudoinstrucțiuni este următorul :

nume	operație	operand	comentarii
poate fi necesar, opțional sau ilegal	**TITLE**		
	ORG		
	EQU		
	SET		
	END		
	IF		
	ENDIF		
	MACRO		
	ENDM		
	DB		
	DS		
	DW		

etopt : **ORG** *expresie* (adresă 16 biți)

Contorul de locații al asamblorului este setat la valoarea expresiei, care trebuie să fie o expresie validă (prin evaluare are o valoare reprezentată pe 16 biți). Următoarele instrucțiuni mașină (sau biți de date generați) sînt asamblate la adresele expresie, expresie+1 etc.

Dacă nici o pseudoinstrucțiune ORG nu apare înaintea primei instrucțiuni mașină, asamblarea programului începe de la adresa 0. Eticheta este opțională.

etopt : **DB** *listă*

DB — (Define Byte(s) of Data)

Definește octeți de date la adresa etopt, etopt+1.

Lista poate fi o listă de :

— Expresii aritmetice și logice care evaluate se pot reprezenta pe 1 octet ;

— Șir de caractere ASCII închise între apostroafe și separate prin virgule.

etopt : **DW** *listă*

DW — (Define Word (Two Bytes) of Data).

Definește cuvinte de date la adresele etopt, etopt+2, . . .

Lista cuprinde expresii care evaluate dau valori pe 16 biți.

etopt : **DS** *expresie*

DS — (Define Storage (Bytes))
Rezervă atît spaţiu de memorie (octeţi) cît este specificat de expresie.
Este echivalentă cu **ORG $**+*expresie*
nume **EQU** *expresie*
EQU (Equate)

Simbolul *'nume'* este definit în tabela de simboli cu valoarea rezultată din evaluarea expresiei. Ori de cîte ori este întîlnit de asamblor, simbolul 'nume' este substituit cu valoarea expresiei.

'nume' asignat cu EQU nu poate fi redefinit (se semnalează multiplă definire).

nume **SET** *expresie*

Simbolul *'nume'* este definit cu valoarea expresiei.

Ori de cîte ori este întîlnit de asamblor, simbolul 'nume' este substituit cu valoarea expresiei. Această valoare este folosită pînă cînd se modifică valoarea asignată numelui cu o altă pseudoinstrucţiune SET.

SET este similară cu EQU în afară de faptul că simbolul poate fi redefinit.

etopt : **END** *adresă de start*

Pseudoinstrucţiune END specifică asamblorului sfîrşitul fizic a programului şi adresa de intrare în execuţie a programului (implicit se consideră 0). O singură pseudoinstrucţiune END poate apare într-un program, fiind ultima linie de program asamblată.

TITLE *'SIR CARACTERE ASCII'.*

Pseudoinstrucţiunea TITLE specifică asamblorului antetul de pagină ce trebuie imprimat, la listare, la fiecare pagină nouă.

IF — ENDIF

Sînt pseudoinstrucţiuni ce permit asamblarea condiţionată a unui program.

etopt : **IF** *expresie*
    linii de program
etopt : **ENDIF**

Asamblorul evaluează expresia şi dacă valoarea acestuia este zero instrucţiunile, pseudoinstrucţiunilor şi directivele dintre IF şi ENDIF sînt ignorate, altfel ele sînt asamblate ca şi cum liniile IF şi ENDIF nu ar fi prezente.

Sînt admise construcţii IF-ENDIF care se cuprind.

Exemplu :

```
IF V1
:
IF V2
:
IF V3
.
ENDIF
:
ENDIF
:
ENDIF
```

MACRO — ENDM : Definire la macroinstrucțiuni.

nume **MACRO** *LISTP*

:

etopt : **ENDM**

LISTP — cuprinde o listă de etichete simbolice-parametrii formali ai macro definiției. Lista poate fi formată și din șirul vid. Pseudoinstrucțiunea MACRO nu poate să apară în lista de linii sursă dintre MACRO și ENDM, adică într-o macroinstrucțiune nu se poate defini o altă macroinstrucțiune.

Macroinstrucțiunile reprezintă o modalitate de programare foarte importantă oferită de Macroasamblorul MAC18 care, utilizată cînd este **cazul, facilitează o programare eficientă și o înțelegere ușoară a unui program scris în limbaj de asamblare. O macroinstrucțiune specifică asamblorului că un simbol (nume de macro) care apare în cîmpul de operație al unei instrucțiuni se substituie cu un grup de instrucțiuni mașină. Atît numele de macro cît și instrucțiunile sînt alese de către programator.**

O macroinstrucțiune din punct de vedere al tratării de către asamblor cuprinde 3 faze :

— Faza de definire ;

— Faza de apel (referire) ;

— Faza de expandare.

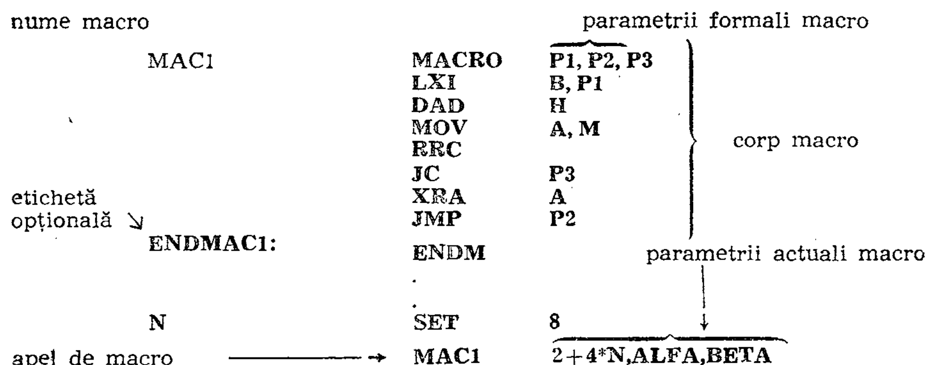
Faza de definire informează asamblorul asupra numelui macro și a secvenței de instrucțiuni ce formează corpul macroinstrucțiunii. Această fază este cerută de utilizator.

Faza de apel (referire) informează asamblorul că este dorită copierea corpului macroinstrucțiunii cu niște parametri dați. Și această fază este cerută de utilizator.

Referirea unei macroinstrucțiuni poate fi făcută în orice punct al unui program, chiar și în interiorul unui alt corp de macroinstrucțiuni.

Faza de expandare nu este prezentă printre liniile sursă ale programului, ea este generată de asamblor la întîlnirea unui apel de macro, generîndu-se liniile din corpul macro cu parametrii actualizați.

Exemplu :



```
linii sursă      LXI    B,00022H
generate de      DAD    H
asamblor         MOV    A, M       substituirea parametrilor formali
                 RRC               cu parametrii actuali
                 JC     ALFA
— expandare      XRA    A
  macro          JMP    BETA
```

Sînt admise construcții de genul apel de macro în corp de macro precum și apeluri recursive cu condiția să se iasă (într-un mod oarecare) din recursivitate.

```
MAC1    MACRO    P1, P2, P3
        .
        .
        .
        MAC2     P2, 21
        .
        .
        .
        ENDM
MAC2    MACRO    PP1, PP2
        .
        .
        .
        ENDM
```

Etichetele definite în corpul unei macro sînt locale (pentru a se evita definirile multiple de etichete) iar la sfîrșitul expandării își pierd valabilitatea.

Pentru ca o etichetă să fie globală (adică să fie recunoscută și în rest, în afară de expandare) trebuie urmată de '::'

```
etichetă     MAC1          MACRO
globală      HERE::
             .
             .
             .
             ENDM
```

După primul apel ea este recunoscută

```
             MAC1
             .
             .
             .
             JMP HERE
```

La al 2-lea apel de macro se va semnala o eroare de multiplă definire pentru eticheta globală.

Utilizarea de etichetă globală este în general folositoare atunci cînd în corpul unei macroinstrucțiuni se definește o rutină care trebuie să apară o singură dată, la prima expandare de macro, dar care trebuie să fie recunoscută în toate apelurile de macro ulterioare.

Condiționarea apariției rutinei la prima expandare se realizează cu ajutorul unei construcții IF-ENDIF.

Exemplu :

```
CV2     SET      0
CONV    MACRO    P1, OUT1
        LXI      D, P1
```

```
            CALL      CV1
            JMP       OUT1
            IF        NOT CV2
CV1:
            DAD       D
            MOV       A, M
            DAA
            ADI       90H
            RET
CV2         SET       —1
            ENDIF
            ENDM
```

La primul apel se generează :

```
            CONV      10, BETA
            LXI       D, 0000AH
            CALL      CV1
            JMP       BETA
CV1:
            DAD       D
            MOV       A, M
            DAA
            ADI       90H
            RET
```

La al 2-lea apel **CONV 21,BETA1** se generează :

```
            LXI       D, 00015H
            CALL      CV1
            JMP       BETA1
```

La fel se va genera la apelurile următoare.

Utilizarea macroinstrucțiunilor este o importantă tehnică de programare care ușurează sarcina programatorului.

8.1.4. Restricții în utilizarea macroasamblorului

Se impun programului utilizator următoarele restricții :

1. Nu sînt permise mai mult de 3 nivele de adîncime în apelurile de macroinstrucțiuni ;
2. Nu sînt permise mai mult de 7 nivele de adîncime de construcții IF-ENDIF, 6 nivele dacă în închiderea blocurilor IF-ENDIF există definiții de macro și 7-n nivele dacă în închiderea blocurilor IF-ENDIF există macroinstrucțiuni care se apelează unele pe altele (n-nivel maxim de adîncime de apel macro $n<3$) ;
3. Nu sînt permise într-o listă definirea a mai mult de 8 parametrii (DB, DW, parametri formali, parametri actuali) ;
4. Nu sînt permise expresii de o complexitate prea mare care necesită introducerea a mai mult de 7 operatori în stiva de operatori (care nu se reduc). De exemplu expresii cu mai mult de 7 '(' care nu se închid ;
5. Nu se pot citi șiruri de caractere care însumate depășesc 60 de octeți ;
6. Nu sînt permise macrodefiniții în macrodefiniții ;
7. Se impune următoarea restricție de memorie :

Considerînd că asamblorul ocupă 7 Ko și că monitorul necesită 1 Ko de memorie RAM, pentru o configurație de 16 Ko se pot scrie programe cu aproximativ 1 000 de simboli iar pentru 64 Ko de memorie, 6 700 simboli, fără utilizarea macroinstrucțiunilor.

Dacă pentru o configurație de 64 Ko se definesc 20 macroinstrucțiuni, apelate în medie de 50 ori, fiecare avînd 2 etichete locale și pînă la 100 de octeți pentru corpul de definire, se pot defini aproximativ 5 200 simboli globali.

8.1.5. Listing, tabelă de simboli, program obiect

Macroasamblorul MAC18 operează cu 3 fișiere de lucru :

— Un fișier de intrare (pe bandă perforată, pe cartele, pe bandă, casetă magnetică) care constă din liniile programului sursă. Fiecare linie se termină cu combinația CRLF (pentru bandă perforată sau magnetică). Dacă fișierul de intrare este pe cartele, fiecare cartelă conține o linie de program.

Liniile de program sursă sînt compuse din cele 4 cîmpuri și sînt despărțite între ele prin unul sau mai mulți separatori. Separator se consideră blancul sau caracterul de tabulare orizontală ;

— Un fișier listing creat de asamblor (pe TTY, imprimantă, bandă magnetică).

Datele sînt destinate unui echipament de listare.

Rolul acestui fișier este de a informa utilizatorul asupra rezultatului asamblării (corectitudine program, generarea codului, liniile de program sursă, tabelă de simboli) ;

— Un fișier de ieșire pentru cod obiect (pe bandă perforată, bandă magnetică).

Fișierul cod obiect — în format hexazecimal — constă din conținutul memoriei programului utilizator care rezultă în urma asamblării, reprezentat în format HEXA-ASCII.

Fișier listing

Se poate suprima pe porțiunile cuprinse între directivele $P=0 și $P=1 sau $P=0 și END.

Listarea programului are loc la trecerea a doua.

Informațiile cuprinse în fișierul de listare sînt grupate în următorul format :

Coloana	Descriere
1	— Cod de eroare. Dacă asamblorul întîlnește o eroare de sintaxă în linia sursă curentă, codul erorii reprezentat printr-o singură literă va apare în această coloană. Altfel în această coloană se va scrie blanc.
2	— Blanc.
3	— Nivel de adîncime al apelului de macroinstrucțiune. Dacă asamblorul expandează o macroinstrucțiune, această coloană va conține nivelul de apel al macroinstrucțiunii (nivelul apelurilor de macro în apeluri de macro).

Altfel această coloană va fi blanc.
4 — Blanc.
5— 8 — Valoarea curentă a contorului de program (PC).
Adresa asignată primului octet al codului obiect din această linie este tipărită în hexazecimal. Suplimentar, în acest cîmp vor apare, valorile generate de pseudoinstrucțiunile ORG, EQU, SET.
9 — Blanc.
10— 11 — Primul octet al codului obiect.
În aceste coloane se tipărește primul octet al codului obiect produs de asamblor pentru linia sursă curentă. Listarea se face în format hexazecimal.
Dacă linia sursă nu produce cod obiect (comentariu, pseudo-instrucțiune) acest cîmp va fi blanc.
12— 13 — Al 2-lea octet al codului obiect.
Cîmpul va fi blanc dacă linia sursă nu generează cod obiect sau generează numai un octet de cod obiect.
13— 14 — Al 3-lea octet al codului obiect dacă linia sursă produce cod pe 3 octeți, altfel va fi blanc.
15— 16 — Al 4-lea octet de cod obiect dacă este generat de pseudo-instrucțiunile DB, DW, altfel va fi blanc.
17 — Indicator de semnalizare operație cu macro. Un '+' care apare în această coloană indică faptul că linia de program sursă face parte dintr-o macrodefiniție sau este rezultatul unei expandări macro.
18 — Blanc.
19—132 — Cîmp de listare text program sursă.

Tabela de simboli

La sfîrșitul fișierului listing se afișează tabela de simboli definiți de utilizator.

Afișarea tabelei de simboli se poate suprima cu directiva $S=0.

Structura tabelei de simboli

— Sînt afișați mai întîi, în ordine alfabetică, simbolii globali definiți de utilizator ;

— Dacă în expandări macro există definiri de etichete locale se afișează simboli definiți local în ordine alfabetică pentru fiecare expandare.

Formatul tabelei de simboli

Prima linie conține textul „**SIMBOLI GLOBALI**", următoarele linii conțin etichete ordonate alfabetic cu valorile asignate lor.

Coloana	Descriere
1— 5	— Caracterele alfanumerice ale primului simbol.
6— 8	— Blanc.
9—12	— Valoarea asignată primului simbol (în hexazecimal).
13—17	— Blanc.
18—22	— Caractere alfanumerice simbol 2.
23—25	— Blanc.
26—29	— Valoarea asignată simbolului 2.

30—34 — Blanc.
35—40 — Caractere alfanumerice simbol 3.
41—42 — Blanc.
43—46 — Valoarea asignată simbolului 3.
47—51 — Blanc.
52—56 — Caractere alfanumerice simbol 4.
57—59 — Blanc.
60—63 — Valoarea asignată simbolului 4.

Pentru fiecare expandare macro cu definire de simboli se listează o tabelă care conține pe prima linie textul :

„SIMBOLI LOCALI EXPANDARE MACRO"

iar în rest o tabelă identică cu prima.

Fișierul de ieșire cod obiect

Formatul fișierului de ieșire cod obiect este hexazecimal.

Fișierul cod obiect hexazecimal conține informația dispusă în blocuri, fiecare bloc conținînd o adresă de încărcare, un contor de lungime și o sumă de control pentru a se verifica corectitudinea încărcării.

Pentru fiecare octet de date (cod) se perforează 2 caractere ASCII care reprezintă decodificarea hexazecimală a octetului respectiv.

Deși o înregistrare hexazecimală este de două ori mai lungă decît o înregistrare în format binar, acest format oferă unele avantaje :

— O bandă cu cod obiect în format hexazecimal poate fi ușor vizualizată și decodificată ;
— Blocurile de înregistrări se pot depista ușor deoarece sînt separate între ele printr-un semn special ':' ;
— În caz de accident al benzii (rupere, deteriorare) se pot depista ușor blocurile distruse, cu adresele lor ;
— Utilizatorul își poate perfora singur o bandă în format hexa (off-line).

Formatul fișierului de ieșire cod obiect este :

Cîmp	Descriere
1	':' Marcă de început înregistrare ;
2—3	Lungimea înregistrării (2 caractere hexa ASCII) ;
4—7	Adresa de încărcare a înregistrării (4 caractere hexa ASCII) ;
8—9	Tip înregistrare (2 caractere ASCII : 00 — înregistrarea curentă 01 — înregistrare de sfîrșit) ;
10—(10+2n)	n octeți de date (cod) ;
(11+2n)—(12+2n)	suma de control a înregistrării.

Exemplu :

```
: 100070007A2F577B2F5F132100003E11E519DH282
: 1000800000E3E1F579174F7817477D176F7C176775
: 0D009000F13DC20C00B77C1F5701F5FC96A2
                ↓                   ↓
          octeți de date      suma de control

: 00000001
  | | | ↓ tip înregistrare
  | | ↓ adresa înregistrării
  | ↓ lungimea înregistrării
  ↓
  marca început de înregistrare
```

8.1.6. Directive pentru controlul asamblării

Programatorul poate comunica asamblorului opțiunile sale în procesul asamblării prin directive care au efect imediat asupra procesului asamblării.

Sintaxa directivelor este :

$literă=număr zecimal

$L=n 1⩽n⩽80 indică marginea stînga a liniei de program sursă ;

Implicit $L=1.

Atenție ! $L=n are efect și asupra directivelor următoare !

Presupunînd apariția la un moment dat printre liniile programului sursă a directivei $L=20, pentru ca o directivă ulterioară să fie luată în considerare ea trebuie să înceapă în coloana 20.

$R=n 1⩽n⩽80 indică marginea dreaptă a liniei de program sursă ;
Implicit $R=80.

$P=n Implicit $P=1 ;
n=0, 1 ;
n=0 indică suprimarea listării pînă cînd se întîlnește $P=1 sau END ;
n=1 indică reluarea listării ;
$P=1 este inefectivă dacă înaintea ei nu a apărut nici o directivă $P=0.

$S=n Implicit $S=1 ;
n=0 indică suprimarea listării tabelei de simboli ;
n=1 indică tipărirea tabelei de simboli.

$W=n 1⩽n⩽132 ;
Indică lățimea listingului obținut în urma asamblării ;
Implicit $W=132.

$H=n Implicit n=1 ;
n=0 suprimă *'perforarea'* benzii obiect ;
n=1 indică reluarea *'perforării'* benzii obiect ;
$H=1 este inefectivă dacă înaintea ei nu a apărut nici o directivă $H=0.

$1=1 Listing la pasul 1 ; Implicit este 0.

8.1.7. Tratarea erorilor

Erorile conținute de un program pot fi împărțite în două clase : erori de fond și erori de sintaxă.

Erorile de fond constau în faptul că algoritmul sau datele sînt eronate sau incomplete. Din punct de vedere al asamblării este indiferent dacă algoritmul în sine este greșit sau numai descrierea lui este eronată.

Erorile de sintaxă constau în nerespectarea regulilor impuse de limbaj la scrierea cuvintelor, expresiilor, frazelor și programelor.

Limbajul de asamblare are un număr finit de reguli de scriere. Aceste reguli care formează sintaxa limbajului trebuie incluse în construcția asamblorului astfel ca acesta să poată detecta orice nerespectare a lor. Ca atare erorile de sintaxă vor fi semnalate la asamblare. Erorile de fond nu

pot fi detectate de compilator decît în măsura în care semantica se reflectă în sintaxă. Ele pot fi detectate și corectate numai prin mai multe încercări de execuție a programului. Din aceste motive, pentru a avea un program corect în majoritatea cazurilor sînt necesare mai multe asamblări și execuții ale programului.

În afara traducerii, asamblorul mai are și sarcina importantă de a ajuta cît mai mult pe utilizator să-și pună la punct programele elaborate. Trebuie ca la sfîrșitul asamblării să se genereze un program în limbaj mașină corect din punct de vedere sintactic. Dacă nu poate face aceasta, asamblorul trebuie să informeze programatorul în modul cel mai clar și mai precis despre toate erorile de sintaxă din programul său. Asamblorul în timpul traducerii textului sursă face și analiza sintactică, verificînd fiecare element dacă respectă toate restricțiile impuse. Cînd una din restricții nu este respectată s-a detectat o eroare, iar asamblorul va genera un text numit indicator de eroare care va conține toate informațiile necesare elaborării mesajului pentru informarea utilizatorului.

8.1.8. Mesaje de eroare

Din considerente de memorie (programul asamblor rulînd pe un microcalculator) mesajele de eroare sînt compuse dintr-o singură literă care apare la fiecare linie eronată. Dacă aceeași linie de program conține mai multe erori atunci prima eroare întîlnită este cea semnalată.

Coduri de eroare

B (Balance Error):

Semnifică neînchiderea corectă de paranteze sau de apostroafe într-o expresie. De exemplu :

MVI H,3* (2+4).
DB 'A"

D (Directive Error):

Directivă incorectă. Poate fi cauzată de o literă eronată pentru nume de directivă sau de argument incorect (prea mare, într-o bază diferită de 10).

Exemplu :

```
$L=120        ;ARGUMENT>80
$U=0          ;U NU ESTE NUME DE DIRECTIVĂ
$W=12H+BETA   ;NU SÎNT ADMISE CA ARGUMENT DECÎT NUMERE ÎN
              BAZA 10
$L120         ;SINTAXA INCORECTĂ, LIPSEȘTE '='
```

E (Expression Error):

Indică o expresie cu construcție eronată. În mod uzual eroarea apare datorită unui operator lipsă, unei virgule omise, sau a unei succesiuni incorecte de operatori.

Exemplu :

```
MVI A,4   CONTOR DE OCTEȚI
          lipsește ';'
MOV H,M   ;COD DE OPERAȚIE INCORECT
```

F (Format Error):

Indică o eroare în formatul instrucțiunii (lipsa unui operand sau prea mulți operanzi).

Exemplu :

```
MOV,D         ;LIPSEȘTE 1 OPERAND
MOV A, B, C   ;1 OPERAND ESTE ÎN PLUS
```

I (Illegal Character):

Indică prezența unui caracter ASCII nepermis într-o linie de program sursă (în contextul respectiv).

Exemplu :

```
MVI H,02B  ;2 NU E CIFRA BINARĂ
MVI H,79Q  ;9 NU E CIFRA OCTALĂ
```

M (Multiple Definition):

Indică definirea de mai multe ori a aceluiași simbol.

Exemplu :

```
ET1: DB 1, 2, 33H
:                  ;SIMBOLUL ET1 A FOST DEJA DEFINIT
ET1: MOV C,M       ;CNZ ESTE PREDEFINIT ÎN ASAMBLOR
CNZ: LXI D,-3      ;(COD DE OPERAȚIE) ȘI NU SE MAI POATE DEFINI
```

N (Nesting Error):

Indică o eroare de imbricare în construcții de genul IF-ENDIF, MACRO-ENDM.

Exemplu :

```
IF D01
LXI D,CHAR
DAD H
ENDIF
XVI A,0
ENDIF          ;NU ÎNCHIDE UN IF
```

P (Phase Error):

Valoarea atribuită unui simbol în faza 1 nu mai coincide cu valoarea calculată pentru simbol în faza a 2-a.

Exemplu :

```
     ORG PHASE ;ETICHETA PHASE NEDEFINITĂ ÎN TRECEREA 1
ET1:           ;LA ACEASTA ETICHETA SE VA
PHASE EQU 99H  ;SEMNALA P ÎN FAZA A 2 A
```

Q (Questionable Syntax):

Indică sintaxa ambiguă a unei linii de program de obicei generată de lipsa codului unei operații.

R (Register Error):

Un registru specificat pentru o operație este invalid pentru acea operație.

Exemplu :

```
PUSH A    ;A NU ESTE REGISTRU PERECHE
DCR 9     ;NUMAR REGISTRU >7
LDAX H    ;TRANSFER DATE INDEXAT NU SE POATE FACE DECÎT PRIN
          ;B&C SAU D&E
```

S (Stack overflow):

Indică depășirea capacității stivelor interne de lucru ale asamblorului (șir prea mare de caractere alfanumerice, expresii prea complicate, prea multe apeluri de macro în alte apeluri de macro, prea multe construcții IF-ENDIF care se cuprind).

Exemplu :

IF VAR1 va genera o eroare S
IF VAR2
IF VAR3 (depășire capacitate IFSTACK)
IF VAR4
IF VAR5
IF VAR6
IF VAR7
.
.
.
ENDIF
.
.
.

T (Table Overflow) :

Indică depășirea dimensiunilor tabelelor afectate (prea multe etichete definite, apeluri de macroinstrucțiuni numeroase, corpuri de macroinstrucțiuni cuprinzînd multe instrucțiuni).

U (Undefined Identifier) :

Indică referirea unui simbol nedefinit.

V (Illegal Value) :

Indică faptul că valoarea unui operand sau a unei expresii depășește dimensiunea corectă pentru acea operație particulară.

Exemplu :

```
DB 291      ;291 NU SE POATE REPREZENTA
            ;PE UN OCTET
```

8.1.9. Încărcarea și lansarea în execuție a macroasamblorului

Macroasamblorul este rezident pe bandă perforată, sau bandă magnetică (bandă sau casetă).

Dacă macroasamblorul este rezident pe bandă perforată se procedează astfel :

— Se lansează monitorul, dacă acesta nu este deja în execuție, după procedura descrisă la capitolul de utilizare a monitorului ;

— Se pregătește cititorul de bandă, se introduce banda în cititor ;

— Se asignează cititorul de bandă ca cititor al sistemului (dacă este cazul) :

.AR=**P**<CR> ;

— Se citește în memorie macroasamblorul ;

.R<CR> :

— După terminarea operației de citire a benzii se fac asignările dorite pentru operația de asamblare. Dacă se lucrează cu benzi magnetice, se pregătesc și se fac deschiderile de fișiere pentru citire/scriere, după caz ;

— Se lansează în execuție macroasamblorul :

.G40<CR>

Macroasamblorul răspunde la consolă cu mesajul :

MACRO ASAMBLOR MAC18Vx.x

P=

Macroasamblorul așteaptă ca utilizatorul să introducă una din cifrele 0, 1, 2, 3 sau 4. Orice alt caracter este ignorat. Dacă macroasamblorul este rezident pe bandă magnetică se pregătește banda și se procedează astfel :

— Se introduce banda cu programul în unitatea dorită și se face asignarea corespunzătoare :

.AR=K sau **.AR=M ;**

— Se deschide pentru citire fișierul cu numărul "n" de pe unitatea 'm' care conține macroasamblorul — programul obiect :

.KIm, *n*<CR> Monitorul va căuta pe unitatea 'm' fișierul "n" ;

— Se citește în memorie macroasamblorul

.R<CR> ;

— Se închide fișierul de intrare (această operație nu este întotdeauna necesară).

.KC<CR>, sau o comandă de închidere selectivă.

În continuare se procedează ca în cazul în care macroasamblorul este rezident pe bandă perforată.

Cu toate că MAC18 este un asamblor cu două treceri, o asamblare se poate efectua în doi sau mai mulți pași, identificați prin cifrele 1, 2, 3 și 4.

Prima trecere este obligatorie și este echivalentă cu pasul 1. A doua trecere poate consta din pașii 2, 3 sau 4.

Acțiunile corespunzătoare fiecărui pas sînt :

P=1 Se formează tabela de simboli.

P=2 Trecerea a doua constă în trimiterea la echipamentul asignat ca LIST a listingului programului asamblat. Nu se generează fișierul obiect.

P=3 Se generează programul obiect la PUNCH dar nu se generează listingul programului.

P=4 Se realizează acțiunea conjugată a pașilor 2 și 3.

P=0 Revenire în monitor (fără reinițializarea monitorului). Același efect se obține prin întrerupere de la panoul frontal.

Avînd în vedere posibilitatea executării repetate a pașilor 2, 3, 4 fără o repetare a pasului 1, precum și inutilitatea generării programului obiect pentru un program sursă cu erori, executarea pasului 2 sau 3 duce la o utilizare mai eficientă a sistemului.

Cu anumite configurații pentru I/E nu se poate executa pasul 4 și deci, faza a doua a asamblării se va desfășura în pașii 2 și 3, de exemplu dacă se fac asignările, .AL=T și .AP=T. *Reamintim că utilizarea a două casete sau a două benzi în paralel, impune restricția ca una să fie utilizată pentru citire, alta pentru scriere.* De asemenea trebuie notat că execuția fiecărui pas necesită citirea textului sursă.

Exemple de succesiune a operațiilor pentru asamblarea unui program sursă în diferite condiții :

— Pe o bandă se găsesc, în ordine, programele obiect MON18, MAC18 și EXT18. În continuare pe această bandă se va introduce fișierul obiect al programului asamblat.

Programul sursă se găsește pe cartele perforate.

.AR=M Banda cu cele trei programe obiect a fost introdusă în unitatea 0.

.KI0,1 Localizează și pregătește pentru citire fișierul 1(MAC18), pe unitatea 0.

.R Citește în memorie, de pe bandă, MAC18.

.KC Închide fișierul deschis pentru citire.
Operația este obligatorie deoarece urmează să se deschidă pentru scriere un fișier pe această bandă.

.AR=C Se introduc cartele în cititor.
.AP=M Programul obiect se va introduce pe bandă.
.KO0,3 Deschide pentru scriere următorul fişier.
.KN *nume fişier*
Se scrie numele fişierului obiect (cel mult 10 caractere).
.G40 Lansează în execuţie asamblorul.
P=1
P=4 S-au reintrodus cartele în cititor pentru pasul 4.
P=0 Reîntoarcere în monitor.
.KC Închide fişierul de ieşire care conţine programul obiect.

Pentru executarea programului asamblat se procedează astfel :

.AR=M
.KI0,3
.R Dacă pseudoinstrucţiunea END din program a specificat adresa de start, programul va fi lansat în execuţie după încărcare, dacă nu, se refac asignările (dacă este cazul) şi se lansează în execuţie programul prin comanda :
.G *ADRESA START.*

În următorul exemplu se consideră că pe o bandă se găsesc în ordine programele MON18, ETX18 şi programul sursă care trebuie asamblat.

Programul obiect va fi introdus într-un fişier, pe o casetă, cu mai multe fişiere, din care nu ne interesează decît primele 4.

.AR=M Banda cu MAC18 şi programul sursă s-a montat.
.AP=K în unitatea 0 ; Programul obiect va fi trimis pe caseta 1.
.KI0,1
.R S-a citit în memorie MAC18.
.KI0,3 Se deschide pentru citire fişierul cu programul sursă.
.KO1,4 Se deschide pentru scriere fişierul al cincilea de pe caseta 1.
.KN *nume* Se atribuie un nume fişierului obiect.
.G40
P=1
P=0 Prin pasul 0 sau întrerupere de la panoul frontal se revine în monitor.
.KI0,3 Se deschide din nou pentru citire fişierul cu programul sursă.
.G
P=4
P=0 Revenire în monitor la terminarea asamblării.
.KC Închide toate fişierele deschise.

8.2. Microasamblorul ASM80

8.2.1. Prezentare generală

Macroasamblorul ASM80 are o serie de facilităţi suplimentare faţă de MAC18, dar şi cerinţe sporite privind configuraţia minimă care permite utilizarea lui. Astfel, ASM80 se execută sub SFDX şi este rezi-

dent pe disc iar pe parcursul asamblării se utilizează fișiere intermediare ce sînt de asemenea localizate pe disc. Configurația minimă de memorie RAM este de 32 Ko.

Una dintre caracteristicile principale ale lui ASM80 este posibilitatea de a cere generarea codului obiect în format realocabil. Aceasta facilitează programarea modulară și oferă o mare elasticitate privind configurația de memorie RAM și PROM a sistemului pentru care se dezvoltă programele.

Alte îmbunătățiri față de MAC18 se referă la operatori și expresii, macroinstrucțiuni, interfața cu utilizatorul și posibilitatea de listare a referințelor interinstrucțiuni. Posibilitatea de generare a codului în format realocabil permite combinarea modulelor programate în limbaj de asamblare cu module în PL/M sau FORTRAN.

În cele ce urmează se vor prezenta numai diferențele față de MAC18 ceea ce implică parcurgerea paragrafului 8.1 chiar dacă se utilizează numai ASM80.

Alfabetul utilizat de ASM80

Setul de caractere recunoscut de ASM80 este format din :

— Litere de la A la Z atît mari cît și mici. Intern sînt considerate toate mari, dar sînt listate exact cum au fost introduse în textul sursă ;
— Cifrele de la 0 la 9 ;
— Caractere speciale :
spațiu + — * / ' () , & : $? = <> % ! ; . CR FF HT ;
— În plus față de acestea, orice caracter ASCII poate apărea în text într-un comentariu sau ca date imediate între apostroafe.

Delimitatori

Unele caractere sau perechi de caractere sînt utilizate ca delimitatori în cadrul textului sursă, după cum urmează :

spațiu	Separator de cîmpuri sau terminator de simbol ;
,	Separator de operanzi în cîmpul operand ;
'...'	Delimitează un șir de caractere ;
(...)	Delimitează o expresie sau subexpresie ;
CR	Terminator de linie ;
HT	Separator de cîmpuri sau terminator simbol ;
;	Delimitator de cîmp comentariu ;
:	Delimitator de simbol utilizat ca etichetă ;
&	Delimitator pentru concatenarea parametrilor formali sau șablon de text macro ;
...	Delimitează lista de parametri sau un parametru care conține în textul său virgule sau spații ;
%	Delimitează un parametru macro ce trebuie evaluat înainte de substituție ;
!	Literalizează următorul caracter care altfel ar fi interpretat ca delimitator ;
;;	Delimitează un comentariu în macrodefiniție ce urmează să fie suprimat la expandare.

Simbolii recunoscuți de ASM80 pot fi formați din 1 pînă la 6 caractere alfanumerice începînd cu literă , ? sau @ , spre deosebire de MAC18

unde pot fi maxim 5 caractere. Atributele asociate simbolilor s-au îmbogățit astfel că simbolii pot fi : locali-globali, interni-externi, absoluți-realocabili.

ASM80 este un asamblor în două faze ca și MAC18 deci se conformează restricțiilor prezentate în paragraful precedent. ASM80 poate fi utilizat și pentru translatarea programelor scrise pentru microprocesorul 8085.

8.2.2. Operatori și expresii

În plus față de operatorii aritmetici, logici și de deplasare utilizați de MAC18, expresiile ce specifică operații valabile la nivelul asamblării programului pot conține și operatori relaționali și de extragere a unui octet dintr-un cuvînt.

Aceștia sînt :

EQ	Egalitate ;
NE	Diferit ;
LT	Mai mic decît ;
LE	Mai mic sau egal ;
GT	Mai mare decît ;
GE	Mai mare sau egal ;
NUL	Operator pentru a testa lipsa parametrilor unei macro.

Operatorii relaționali se utilizează ca și cei de deplasare și logici : *operand1* **operator** *operand2*. Operatorul trebuie separat de fiecare operand prin cel puțin un spațiu. Rezultatul unei operații relaționale este adevărat dacă este satisfăcută operația, altfel este fals (adevărat se reprezintă cu 1 peste tot iar fals cu 0 peste tot). Comparația se face fără să se țină seama de semn considerîndu-se biții operanzilor ca atare. Astfel un număr negativ este întotdeauna mai mare decît unul pozitiv. Operatorul NUL va fi prezentat în legătură cu macroinstrucțiunile în paragrafele următoare.

HIGH	Extrage biții 8 : 15 (partea mai semnificativă) a unei valori de 16 biți ;
LOW	Extrage biții 0 : 7 (partea mai puțin semnificativă) a unei valori de 16 biți.

Trebuie notat că ASM80 consideră toți simbolii externi și cei realocabili cu valori pe 16 biți. Astfel, dacă se utilizează astfel de simboli în expresii pentru calculul operanzilor imediați reprezentați pe 8 biți trebuie utilizați operatorii HIGH sau LOW, altfel asamblorul consideră implicit operatorul LOW și emite un mesaj de eroare.

De asemenea segmentele ce conțin simboli asupra cărora se aplică operatorul HIGH trebuie alocate în memorie în limitele unei pagini utilizînd opțiunea PAGE și DSEG sau CSEG. Altfel s-ar putea ca la realocare să apară un transport din octetul mai puțin semnificativ spre cel mai semnificativ care, fiind ignorat, va duce la un rezultat greșit.

Prioritățile operatorilor sînt următoarele :

1. Expresii în paranteze (prioritate maximă)
2. NUL Acesta apare în plus față de MAC18

3. HIGH, LOW. Nu apar la MAC18
4. *,/, MOD, SHL, SHR
5. +, — (unari sau binari)
6. EQ, LT, LE, GT, GE, NE. În plus față de MAC18
7. NOT
8. AND
9. OR, XOR.

Se reamintește utilizatorilor că operatorii logici, relaționali, de deplasare, MOD, NUL, HIGH, LOW trebuie separați de operanzi prin cel puțin un spațiu.

Dacă în expresii se utilizează operatori unari, se vor utiliza paranteze pentru a evita doi operatori alăturați.

Expresii realocabile

Pentru programele realocabile există unele restricții privind omogenitatea expresiilor relativă la tipul simbolilor din expresie.

Într-un program absolut (asamblat cu ASEG) toți simbolii sînt absoluți.

Următorii simboli sînt realocabili :

— Etichetele din modulele asamblate cu CSEG sau DSEG ;
— Simbolii echivalați prin SET, EQU cu expresii realocabile ;
— Simbolii STACK și MEMORY predefiniți de ASM80 ;
— Simbolii externi ;
— Valoarea contorului de alocare ($) în modulele asamblate cu opțiunea CSEG sau DSEG.

Dacă se notează cu A atributul absolut și cu R atributul realocabil iar cu I o combinație ilegală, se obțin regulile de compunere a expresiilor și tipul rezultatului, funcție de tipul operanzilor, prezentate în continuare :

Operator			X=A ; Y=A	X=A ; Y=R	X=R ; Y=A	X=R ; Y=R
X	+	Y	A	R	R	I
X	—	Y	A	I	R	A
X	*	Y	A	I	I	I
X	/	Y	A	I	I	I
X	MOD	Y	A	I	I	I
X	SHL	Y	A	I	I	I
X	SHR	Y	A	I	I	I
X	EQ	Y	A	I	I	A
X	LT	Y	A	I	I	A
X	LE	Y	A	I	I	A
X	GT	Y	A	I	I	A
X	GE	Y	A	I	I	A
X	NE	Y	A	I	I	A
X	AND	Y	A	I	I	I
X	OR	Y	A	I	I	I
X	XOR	Y	A	I	I	I
	NOT	X	A	A	I	I
	HIGH	X	A	A	R	R
	LOW	X	A	A	R	R
	+	X	A	A	R	R
	—	X	A	A	I	I

Se observă că singurele operații binare permise cu operanzi realocabili sînt scăderea și operațiile de comparare (relaționale). Dacă unul din

simboli este extern, implicit el este considerat realocabil şi singurele operaţii acceptate sînt cele care produc rezultat realocabil:
extern±*absolut*
absolut+*extern*
HIGH *extern*
LOW *extern*
\+ *extern*.

8.2.3. Directive de asamblare

Pseudoinstrucţiunile EQU, SET, DB, DW, DS, END se utilizează ca şi la MAC18. Lista de date specificate de DB şi DW poate să conţină pînă la 8 elemente separate prin virgulă. Expresiile se conformă regulilor enunţate pentru ASM80.

Pentru asamblare condiţionată se utilizează directivele:

[*etopt:*] IF *expresie*
linii de program
[*etopt:*] ELSE]
linii de program
[*etopt:*] **ENDIF**

Expresia asociată lui IF trebuie să conţină numai simboli definiţi în prealabil. ELSE este opţională, deci se pot utiliza construcţii în forma IF-ENDIF. Dacă se utilizează ELSE trebuie notat că o singură directivă ELSE poate să apară într-un bloc IF-ENDIF. Dacă bitul 0 al valorii expresiei este 1 se asamblează liniile de program cuprinse între IF şi ELSE iar cele cuprinse între ELSE şi ENDIF sînt ignorate. Dacă bitul 0 al expresiei evaluate este 0 se ignoră blocul între IF-ELSE şi se asamblează cel dintre ELSE şi ENDIF. Sînt admise blocuri IF-ENDIF care se cuprind pînă la 8 nivele. IF-ENDIF pot să apară în macrodefiniţii şi de asemenea, în blocuri IF-ENDIF pot apare macrodefiniţii. Trebuie avute în vedere precauţii privind definirea simbolilor şi macroinstrucţiunilor în blocuri IF-ENDIF pentru a evita erorile de simboli nedefiniţi, definiri de macro care nu se termină cu ENDM.

În legătură cu directiva END trebuie notat că în cazul combinării mai multor module realocabile, un singur modul denumit modul principal poate să conţină END cu argument, reprezentînd adresa de start a programului.

Pentru a permite asamblarea unor programe sursă formate din mai multe benzi perforate se poate utiliza directiva EOT cu următorul format:

[*etopt:*] **EOT**

La întîlnirea în textul sursă a acestei directive se emite la consolă mesajul **NEXT TAPE** şi se opreşte procesul de asamblare pînă ce se introduce de la consolă caracterul spaţiu (blanc), bineînţeles după ce s-a montat în cititor următoarea bandă.

Directive pentru controlul realocării

Scrierea programelor realocabile facilitează programarea modulară permiţînd asamblarea şi punerea la punct a programelor în mai multe etape, pe module şi de asemenea facilitează gestiunea memoriei în special

cînd se utilizează diferite configurații de memorie RAM și ROM. Un program poate fi asamblat în trei segmente, fiecăruia fiindu-i asociat un contor de alocare separat.

— Segment de cod absolut (ASEG) ;
— Segment de cod realocabil (CSEG) ;
— Segment de date (și/sau cod) realocabil (DSEG).

De fapt ASEG, CSEG și DSEG arată contorul de alocare ce va fi utilizat în continuare iar ASM80 produce un modul format din 4 segmente : cod, date, stivă și memorie disponibilă indiferent de numărul de apariții a acestor directive. În continuare se prezintă formatul și efectul acestor directive.

[*etopt:*] **ASEG.**

Instrucțiunile care urmează pînă la noua directivă CSEG, DSEG sau END vor fi asamblate în mod absolut. Contorul de alocare este implicit egal cu 0 și poate fi modificat cu directiva ORG. Directiva ASEG se consideră implicit și dacă nu se specifică altă directivă de alocare se generează un program obiect direct executabil.

[*etopt:*] **CSEG** {*nimic* | **PAGE** | **INPAGE**}

Instrucțiunile care urmează pînă la o nouă directivă de alocare vor fi asamblate în mod realocabil utilizînd contorul de alocare asociat segmentului de cod realocabil. Semnificația operandului este următoarea :

— nimic : Lipsa operandului arată că, implicit, se face asamblarea începînd cu următorul octet de memorie disponibilă ;
— PAGE : Segmentul de cod ce urmează trebuie alocat de către LOCATE la o adresă multiplu de 256 ;
— INPAGE : Segmentul de cod ce urmează trebuie plasat, la realocare, în aceeași pagină de 256 octeți.

Dacă se utilizează mai multe directive CSEG într-un program toate trebuie să aibă același argument.

[*etopt:*] **DSEG** {*nimic* | **PAGE** | **INPAGE**}

Instrucțiunile și datele care urmează vor fi asamblate în mod realocabil utilizînd contorul de alocare asociat segmentului de date. Semnificația operandului este aceeași ca la CSEG. Dacă se utilizează directiva ORG în segmente realocabile, valoarea argumentului poate fi absolută sau realocabilă în segmentul respectiv.

Directive pentru editarea legăturilor

Pentru a permite referirile intermodule în cazul programelor realocabile (sau absolute) formate din mai multe module au fost prevăzute directive prin care se specifică aceste referințe. Acestea sînt : PUBLIC, EXTRN, NAME, STKLN.

[*etopt:*] **PUBLIC** *lista de nume*

Simbolii din lista de nume, separați prin virgule, sînt definiți în modulul curent și sînt accesibili din alte module sau programe. Directiva

PUBLIC poate să apară oriunde în cadrul programului. Simbolii din lista de nume trebuie să fie definiţi o singură dată.

[*etopt:*] **EXTRN** *listă de nume*

Simbolii din lista de nume, separaţi prin virgule, sînt definiţi în alte module dar sînt referiţi şi în modulul curent. Directiva EXTRN poate să apară oriunde în program. Dacă lista de nume conţine simboli definiţi în program sau simboli predefiniţi se semnalează o eroare de multiplă definire.

[*etopt:*] **NAME** *nume modul*

Directiva NAME oferă posibilitatea referirii modulului de cod obiect în care apare cu numele definit ca operand (nume modul). Aceasta este necesară atunci cînd se combină mai multe module pentru a forma un program. Ordinea de legare a modulelor se specifică prin numele acestora. Directiva NAME trebuie să apară la începutul modulului, înaintea primei instrucţiuni şi eventual după comentarii sau linii de control.

[*etopt:*] **STKLN** *expresie*

Directiva STKLN permite specificarea numărului de octeţi, ce vor fi rezervaţi pentru stiva modulului curent (valoarea expresiei din cîmpul operand). Avînd în vedere că la editarea legăturilor se va forma o singură stivă pentru tot programul, este suficient ca un singur modul să specifice dimensiunea maximă a stivei cu directiva STKLN. Dacă se face testarea fiecărui modul separat, trebuie dimensionată stiva în fiecare modul. Dacă apar mai multe directive STKLN în modulul curent, doar ultima va fi luată în considerare.

STACK şi **MEMORY**

Acestea sînt cuvinte rezervate care pot fi declarate externe în program ele fiind definite de LOCATE.

Astfel prin STACK se poate face o referire simbolică la adresa de început a stivei (şi la următoarele STACK+1, STACK+2 ...).

Prin MEMORY se pot face referiri simbolice la prima adresă de memorie disponibilă care începe imediat după ultima adresă ocupată de program. În acest fel se facilitează gestiunea spaţiului disponibil, ultima celulă de memorie RAM putînd fi determinată cu o procedură din monitor.

8.2.4. Macroinstrucţiuni

Directivele utilizate pentru definirea şi apelul macroinstrucţiunilor sînt : MACRO, ENDM, EXITM, LOCAL, REPT, IRP, IRPC.

Orice macroinstrucţiune trebuie definită înainte de a fi apelată. Convenţia de specificare a domeniului de valabilitate a unei etichete în corpul macroinstrucţiunii s-a schimbat, astfel că etichetele locale sînt declarate explicit prin directiva LOCAL altfel sînt considerate globale. Dacă o macroinstrucţiune ce conţine etichete globale este apelată de cel puţin două ori, se semnalează eroare de multiplă definire. Numele parametrilor formali au în mod implicit atributul de simboli locali.

nume **MACRO** [*parametri formali*]

Față de cele prezentate la MAC18, trebuie ținut seama de următoarele :

— Se recomandă ca simbolii rezervați să nu apară ca nume în lista de parametri formali. Dacă apar, substituirea cu parametri reali se va face corect, dar valorile acestor simboli nu vor putea fi utilizate în corpul macro. Astfel dacă în lista de parametri formali apare simbolul A, în corpul de definiție nu se va putea utiliza registrul A (acumulatorul) cu semnificația lui de bază.

— Dacă într-un comentariu apar parametri formali, aceștia nu sînt luați în considerare și nu se face substituirea lor cu parametrii reali.

— & Ampersand : Se utilizează pentru prefixarea sau postfixarea parametrilor formali cînd apar ca șiruri de caractere. Este utilizat ca operator de concatenare de texte și parametri formali. La expandare orice caracter & care precede sau urmează imediat un parametru formal este înlăturat și substituția parametrului real apare în acel punct. Dacă nu este adiacent unui parametru formal, & nu este înlăturat și apare ca atare în textul macro după expandare. Nu sînt admise spații între & și parametrul formal sau textul ce trebuie concatenat. Astfel, parametrii formali incluși în șiruri de caractere sînt recunoscuți numai dacă sînt delimitați de caractere &. De asemenea, caracterele & sînt recunoscute ca operatori de concatenare numai dacă sînt adiacente cu parametri formali.

— < > Paranteze unghiulare : Se utilizează pentru delimitarea unui text care conține alți delimitatori (virgulă, spațiu etc.). De exemplu dacă se dorește transmiterea unui parametru real de forma MOV A, M, acesta trebuie scris astfel <MOV A, M> pentru a fi transmis ca atare. Dacă se transmit astfel de parametri la macro care se cuprind, se vor utiliza o pereche de < > pentru fiecare nivel de macro în macro.

— ! Semnul exclamării : Se utilizează ca și caracter de literalizare la transmiterea parametrilor actuali. Caracterul ce urmează după ! este transmis ca atare chiar dacă este un delimitator. Astfel pentru transmiterea, ca parte a unui parametru, a caracterelor, sau ! sau < se va scrie !, sau ! ! sau ! < .CR nu poate fi transmis ca parametru actual.

— ; ; Punct virgulă dublu : Se utilizează înaintea unui comentariu din macrodefiniție care nu trebuie să apară la expandare. Se face astfel o economie de memorie iar comentariul apare, totuși, în listingul programului la definire.

— NUL : Se utilizează pentru a specifica în mod explicit absența unui parametru real la apelul unei macro. Parametrul omis poate fi reprezentat prin doi delimitatori consecutivi (P1,,P3) sau prin două apostroafe consecutive '' ,P1, P2. Dacă într-o macro cu parametri formali W, X, Y, Z poate lipsi fie X fie Y dar nu ambii, se poate utiliza NUL pentru a detecta eroarea :

IF NUL X&Y
EXITM

Alte diferențe față de MAC18 vor fi prezentate în continuare. Trebuie notat că directiva ENDM nu admite etichetă sau operand.

LOCAL *listă de nume*

Simbolii specificați în lista de nume au valabilitate numai în expandarea curentă. La fiecare expandare se atribuie fiecărui simbol local un nume unic de forma ??nnn începînd cu ??001.

Parametrii formali nu pot apare în directiva LOCAL deoarece aceştia sînt în mod implicit locali. Directivele LOCAL trebuie să preceadă prima instrucţiune din corpul de definiţie.

[*etopt:*] **REPT** *expresie*

Blocul cuprins între REPT şi ENDM va fi repetat de un număr de ori egal cu valoarea expresiei din cîmpul operand. Expresia trebuie să fie definită în prealabil. Expandarea se face în momentul întîlnirii directivei REPT şi deci nu este necesară o apelare explicită. Pentru rotirea acumulatorului de 6 ori se va scrie :

```
RRC6    REPT    6
        RRC
        ENDM
```

O altă macroinstrucţiune implicită ce nu există la MAC18 este :

[*etopt:*] **IRP** *parametru, <listă>*

Directiva IRP permite o repetare indefinită a blocului dintre IRP şi ENDM în felul următor :

În cîmpul operand se specifică un parametru formal şi o listă de parametri reali delimitată prin paranteze unghiulare. Se ia primul parametru real din listă şi se expandează macroinstrucţiunea, apoi se ia cel de al doilea parametru şi făcînd substituţia parametrului formal (peste tot unde apare) se obţine cea de a doua repetare. Se continuă în acest mod pînă la epuizarea listei cu parametri reali. Astfel o secvenţă care preia datele din diferite locaţii de memorie şi le înscrie într-un tablou se poate scrie, cu IRP, astfel :

Definirea :		Codul generat :	
LXI	H,TAB	LXI	H,TAB
IRP	X,<ET1,4F0H,ET3>	LDA	ET1
LDA	X	MOV	M,A
MOV	M,A	INX	H
INX	H	LDA	4F0H
ENDM		MOV	M,A
		INX	H
		LDA	ET3
		MOV	M,A
		INX	H

[*etopt:*] **IRPC** *parametru, text*

Secvenţa dintre IRPC şi ENDM va fi repetată de atîtea ori cîte caractere are text. La fiecare repetare parametrul formal specificat va fi înlocuit cu caracterul corespunzător din text (Primul caracter, al doilea etc). Dacă text este inclus în paranteze unghiulare orice delimitator din text va fi tratat ca un caracter oarecare generîndu-se o expandare şi pentru acesta. Dacă text este nul se generează o expandare înlocuindu-se parametrul formal cu nul. Ex. :

Definirea :		Codul generat :	
LXI	H, DATA	LXI	H,DATA
IRPC	X, 1983	MOV	M,1
MOV	M,X	INX	H
INX	H	MOV	M,9
ENDM		INX	H
		MOV	M,8
		INX	H
		MOV	M,3
		INX	H

[*etopt:*] **EXITM**

Directiva EXITM permite terminarea forțată a expandării unei macroinstrucțiuni (inclusiv REPT, IRP, IRPC) și ignorarea textului dintre EXITM și ENDM. Chiar dacă se utilizează EXITM, este necesar ca definirea să se termine cu ENDM.

Definiții de macro în macrodefiniții

ASM80 permite definirea unei macroinstrucțiuni în interiorul unei alte macrodefiniții. Nu există nici o limitare privind numărul de macrodefiniții în microdefiniții. O macroinstrucțiune este definită și poate fi chemată pentru expandare numai dacă toate macrodefinițiile de pe nivelele mai mari au fost chemate și expandate deja. Apelul unei macro pe un anumit nivel, definește și permite apelul macrodefiniției de pe nivelul imediat inferior. Utilizînd această posibilitate de definire a unei macro în altă macro se poate redefini o macrodefiniție dacă numele macrodefiniției inclusă într-o definiție este parametru formal într-o macro de nivel mai ridicat. La fiecare apelare a unei macro de nivel mai mare, macroinstrucțiunea de nivel mai mic este altfel definită dacă cele două au parametri formali comuni.

Apeluri de macro

Modul de apel și corespondența parametrilor formali și efectivi este la fel ca la MAC18. Sînt permise apeluri de macro în macro (apeluri care se cuprind) pînă la 8 nivele. O macro apelată într-o macrodefiniție nu este obligatoriu să fie definită în momentul definirii macroinstrucțiunii ce o apelează dar trebuie să fie definită în momentul cînd aceasta este apelată.

O macro se poate apela pe ea însăși (apeluri recursive) pînă la 8 nivele.

La apelare parametrii formali sînt înlocuiți cu parametri efectivi ca o substituție de șiruri de caractere. Dacă se dorește evaluarea unui parametru efectiv înainte de expandare, acesta trebuie prefixat (%). Parametrul va fi transmis ca un număr zecimal reprezentînd valoarea acestuia.

Exemple :

1. Să se scrie o macro care rotește acumulatorul la stînga sau la dreapta de un număr de ori dat.

```
ROTLR MACRO X, Y
      IF X EQ 'R'
      REPT Y
      RAR
      ENDM
      ENDIF
      IF X NE 'L'
      EXITM
      ELSE
      REPT Y
      RAL
      ENDM
      ENDIF
      ENDM
```

Apelarea se poate face astfel :

ROTLR 'R',5

ROTLR 'L',%NROT-1

În ultimul caz expresia este evaluată imediat și se transmite ca parametru efectiv valoarea, în zecimal, rezultată.

2. Funcţia de mai sus poate fi realizată astfel :

```
ROTLR   MACRO   X, Y
        REPT    Y
        RA&X
        ENDM
        ENDM
```

Apelul se poate face : **ROTLR R,5** sau
ROTRL L,3

3. Conţinutul locaţiilor SRCi, SRCj,... poate fi mutat la locaţiile DSTi, DSTj,... cu următoarea macroinstrucţiune :

```
MOVE    MACRO   X,Y,Z
        IRPC    PARAM,Z
        LDA     X&&PARAM
        STA     Y&&PARAM
        ENDM
        ENDM
```

Apelul se face astfel : **MOVE SRC, DST, 123**

4. În exemplul următor se arată o macroinstrucţiune care se autotransformă într-o subrutină după prima apelare.

```
MACSUB  MACRO
        CALL    SUBR        ;; redefinirea lui MACSUB
        ENDM
        CALL    SUBR
LINK:   JMP     GAT
SUBR:   .
        .
        .
        RET
GAT:    ENDM
```

În prima apelare se redefineşte MACSUB astfel că următoarele apeluri ale lui MACSUB vor genera CALL SUBR. De asemenea la prima apelare se generează complet codul subrutinei SUBR, se generează CALL SUBR se execută subrutina şi apoi se generează JMP GAT pentru a nu executa din nou SUBR.SUBR trebuie să fie global pentru a putea fi accesibil în afara primei expandări.

5. Utilizînd directivele de asamblare condiţionată se poate obţine o altă soluţie la problema precedentă.

```
TRUE    EQU     0FFH
FALSE   EQU     0
FIRST   SET     TRUE
MACSUB  MACRO
        CALL    SUBR
        IF      NOT         FIRST
        EXITM
        ENDIF
FIRST   SET     FALSE
        JMP     GAT
SUBR:   .
        .
        .
        RET

GAT:    ENDM
```

La prima apelare FIRST = TRUE şi deci se ignoră EXITM din blocul IF-ENDIF astfel că se generează codul pentru subrutină şi pentru apelul acesteia. La următoarele apeluri FIRST = FALSE şi se generază doar CALL SUBR.

6. Instrucţiunea GOTO calculat din limbajele de nivel înalt se poate implementa cu următoarele macrodefiniţii. Secvenţa de apel va fi :

GOTO CASE, <ET1, ET2, ET3, ET4, ET5>

Numărul de etichete din listă este limitat doar de lungimea unei linii de program sursă. Definiţia este :

```
GOTO     MACRO    INDEX, LISTA
         LOCAL    JMPTAB
         MVI      A,INDEX          ; A = Indexul în tabelă
         LXI      H,JMPTAB         ; HL = Adresa tabelei de salturi
         JMPMAC                    ; Apel macro JMPMAC
JMPTAB:  IRP      PARAM, <LISTA>
         DW       PARAM            ; Construieşte tabela de adrese
         ENDM
         ENDM
```

Macro GOTO utilizează pe JMPMAC, care va fi definită în continuare.

```
         JMPMAC   MACRO
         JMPCOD:  ADD      A          ; A = A * 2
                  MVI      D,0        ; D = 0
                  MOV      E,A        ; DE = deplasament
                  DAD      D          ; HL = HL + DE
                  MOV      E,M        ; E = primul octet al adresei de salt
                  INX      H
                  MOV      D,M        ; D = al doilea octet al adresei
                  XCHG                ; HL ↔ DE
                  PCHL                ; Salt la adresa calculată
         JMPMAC   MACRO               ; Redefineşte pe JMPMAC
                  JMP      JMPCOD     ; Pentru economie de cod generat
                  ENDM
                  ENDM
```

JMPMAC este redefinită astfel că doar la prima apelare se generează cod pentru întregul corp de definire, iar la următoarele apelări se generează doar JMP JMPCOD.

8.2.4. Caracteristici de utilizare a macroasamblorului ASM80

ASM80 se execută sub SFDX-18 şi necesită următoarea configuraţie :

— Sistem M18, M118 sau M18B cu minimum 32Ko de memorie RAM sau minim 48Ko dacă programele asamblate conţin macroinstrucţiuni ;
— Cel puţin o unitate de discuri ;
— Imprimantă, dacă se doreşte listarea programului asamblat.

Pentru a putea fi executat pe sisteme cu 32Ko ASM80 este segmentat. Dacă însă programul conţine macroinstrucţiuni sau dacă se doreşte o execuţie mai rapidă se specifică opţiunea MACROFILE (paragraful 8.2.5) şi ASM80 se va executa în mod nesegmentat. Acest mod de execuţie necesită cel puţin 48Ko de memorie RAM.

Numărul de simboli ce pot fi incluşi în program este de aproximativ 200 pentru modul de lucru segmentat şi 32Ko de memorie şi aproximativ 800 pentru 48Ko şi execuţie în mod nesegmentat. Fiecare 16Ko de memorie adiţională creşte cu 2 000 numărul de simboli acceptaţi de ASM80. Macrodefiniţiile sînt memorate pe disc deci utilizarea de macro necesită o zonă mică de memorie pentru păstrarea parametrilor actuali dar această zonă este fixă şi nu creşte proporţional cu numărul şi dimensiunea macrodefiniţiilor. Programele foarte mari pot fi segmentate avînd în vedere că ASM80 generează cod realocabil.

Fişierele utilizate de ASM80

ASM80 utilizează fişiere cu nume predefinite deci utilizatorul trebuie să evite utilizarea acestor nume. Astfel, ASM80 este numele fişierului ce conţine rădăcina asamblorului iar ASM80.nnn unde n = 0,1,2,3,4 sînt segmente ale lui ASM80 şi trebuie să fie toate pe acelaşi disc (în orice unitate). Dacă programul conţine macro se generează un fişier ASMAC. TMP pe unitatea specificată cu opţiunea MACROFILE. Implicit acest fişier este generat pe discul cu programul sursă. Dacă se specifică opţiunea XREF este necesar ca fişierul ASXREF conţinînd segmentul de program ce execută funcţia de afişare a referinţelor interinstrucţiuni să fie rezident pe acelaşi disc cu ASM80. Specificarea acestei opţiuni implică şi generarea unui fişier de lucru ASXREF.TMP pe discul cu programul sursă.

Dacă în linia de comandă prin care se cere o asamblare nu se specifică fişierul obiect şi fişierul listing, se creează pe acelaşi disc cu programul sursă, fişierele sursă. OBJ şi sursă. LST.

Lansarea în execuţie a asamblorului :

Formatul comenzii de lansare a asamblorului este :

ASM80 *fişier-sursă listă-opţiuni* <cr>

fişier-sursă : reprezintă numele fişierului ce conţine programul sursă ;

lista-opţiuni : reprezintă opţiunile specificate explicit, fără prescurtări separate prin cel puţin un spaţiu. Opţiunile implicite nu trebuie specificate (paragraful următor).

Dacă linia de comandă a fost corect specificată, ASM80 emite un mesaj de lansare în execuţie şi începe procesul de translatare a programului sursă conform cu opţiunile specificate în linia de comandă sau în programul sursă (opţiunile cu $).

Pentru a reduce timpul de asamblare se recomandă :

— Opţiunea MACROFILE dacă sistemul are peste 48Ko de RAM ;
— Opţiunea NOOBJECT pentru primele asamblări (cu erori) ;
— NOOBJECT şi NOPRINT afişează la consolă doar numărul de erori detectate la asamblare ;
— NOLIST şi NOSYMBOLS afişează doar liniile eronate ;
— Minimizarea transferurilor cu discul evitînd, ori de cîte ori este posibil, utilizarea opţiunilor INCLUDE, linii de control în programul sursă, macrodefiniţii (în special cele care se cuprind).

8.2.5. Opţiuni de asamblare

Opţiunile de asamblare care controlează diferitele moduri de execuţie a asamblorului, pot fi specificate în două moduri :

— Prin comenzi specificate în timpul asamblării ;
— Prin linii de comandă intercalate în programul sursă, ceea ce permite controlul asamblării pe porţiuni de program.

Dacă comanda specificată în timpul asamblării conţine o opţiune incorectă se ignoră întreaga linie. Dacă o linie de comenzi din programul sursă conţine o opţiune incorectă se ignoră toate opţiunile ce urmează şi cea eronată.

După modul în care acţionează aceste opţiuni ele sînt împărţite în două clase :

— Opţiuni PRIMARE : acestea pot fi specificate o singură dată, pentru o asamblare. Liniile din programul sursă ce conţin opţiuni primare trebuie plasate înaintea primei linii de program sursă (inclusiv comentarii).

— Opţiuni generale : pot fi specificate oriunde, oricînd şi ori de cîte ori.

Mai jos se arată sumarul opţiunilor. Prima opţiune (din cele două alternative, dacă este cazul) este cea implicită.

Opţiunea	**P/G**	**Domeniul de aplicare**
OBJECT/NOOBJECT	P	Fişier obiect
NODEBUG/DEBUG	P	Fişier obiect
PRINT/NOPRINT	P	Fişier listing
COND/NOCOND	G	Fişier listing
LIST/NOLIST	G	Fişier listing
SYMBOLS/NOSYMBOLS	P	Fişier listing
PAGING/NOPAGING	P	Fişier listing
PAGELENGTH (66)	P	Fişier listing
PAGEWIDTH (120)	P	Fişier listing
/EJECT	G	Fişier listing
/TITLE	G	Fişier listing
NOXREF/XREF	P	Referinţe interinstrucţiuni
NOMACROFILE/MACROFILE	P	Macroinstrucţiuni
NOMACRODEBUG/MACRODEBUG	P	Listare macro
GEN/NOGEN	G	Listare macro
/MOD85	P	Programe pentru 8085
/SAVE	G	Salvează opţiunile
/RESTORE	G	Restaurează opţiunile
/INCLUDE	G	Fişiere sursă
NOTTY/TTY	P	Funcţiile consolei.

Prima opţiune este cea considerată implicit de ASM80. Specificarea detaliată a opţiunilor este dată în continuare.

Opţiune P	**Funcţie**
OBJECT (fişier)	Se generează un fişier obiect la *fişier*, specificat în opţiune. Dacă opţiunea lipseşte se consideră implicit OBJECT (*sursă*. OBJ).
NOOBJECT	Suprimă generarea codului obiect.
NOBEBUG	Tabela de simboli nu este inclusă în fişierul obiect.
DEBUG	Dacă s-a specificat generarea programului obiect în fişierul respectiv se include şi tabela de simboli.
PRINT (fişier)	Se generează fişier listing care este transmis la fişier specificat în opţiune. Dacă opţiunea este omisă se consideră implicit PRINT *(sursă*.LST) Vezi şi opţiunea LIST.
NOPRINT	Nu se generează listingul programului.

SYMBOLS	Tabela de simboli este transmisă la fișierul specificat prin PRINT.
NOSYMBOLS	Listarea tabelei de simboli este suprimată.
PAGING	Listingul este paginat, se scrie antetul de pagină la fiecare început de pagină.
NOPAGING	Paginarea este suprimată.
PAGELENGTH (n)	Fiecare pagină conține n linii și n poate fi cel puțin 12 și include 3 linii la începutul paginii, 3 la sfîrșit și liniile de antet. Dacă se specifică $n<12$ se consideră $n=12$. Implicit se consideră PAGELENGTH (66).
PAGEWIDTH (n)	Fiecare linie conține n caractere, cu $72 \leq n \leq 132$. Liniile mai mari de 132 caractere sînt trunchiate iar liniile ce depășesc PAGEWIDTH sînt continuate cu coloana 25 din linia următoare. Implicit $n=120$.
NOXREF	Se suprimă generarea referințelor interinstrucțiuni.
XREF	Se generează referințe interinstrucțiuni ce sînt trimise la fișierul ASXREF.TMP și listate la fișierul specificat prin PRINT.
NOMACROFILE	Nu sînt macrodefiniții în programul sursă. Dacă sînt se vor semnala ca erori. ASM80 se execută în mod segmentat (mai lent).
MACROFILE (*disc*)	Fișierul ASMAC.TMP va fi creat pe discul specificat. Dacă se specifică doar MACROFILE, acest fișier se creează pe același disc cu sursa ASM80 se execută în mod nesegmentat, în cel puțin 48Ko de memorie RAM.
NOMACRODEBUG	Simbolii din macrodefiniții generați de asamblor nu sînt incluși în fișierele listing și obiect.
MOD85	Specifică faptul că programul sursă conține și instrucțiuni 8085. Implicit se consideră sursă pentru 8080.
NOTTY	Nu se simulează 'form-feed'.
TTY	Se consideră ieșire pe TTY și se simulează 'form-feed'

Opțiune G	**Funcție**
COND	Directivele de amplasare condiționată și codul omis prin condiție falsă sînt incluse în fișierul listing dacă s-a selectat opțiunea LIST.
NOCOD	Directivele și codul pentru condiție falsă sînt suprimate din listing, de asemenea și directivele EXITM.
LIST	Se generează un fișier listing de ieșire și este trimis la fișierul specificat prin PRINT.
NOLIST	Se suprimă generarea fișierului listing mai puțin liniile eronate.
EJECT	Salt la începutul paginii următoare, determinat de opțiunea PAGELENGTH.

TITLE (*'sir'*)	Titlul specificat prin sir se listează în linia a doua a antetului de pagină, coloanele 1—64. Şirul este trunchiat la 64 şi trebuie inclus în apostroafe. Are efect pînă la o nouă opţiune TITLE. Şirul specificat trebuie să conţină cel puţin un caracter.
GEN	Textul expandărilor macro se listează la fişierul specificat de PRINT dacă s-a specificat LIST.
NOGEN	Se suprimă listarea expandărilor macro.
SAVE	Opţiunile curente pentru LIST, COND, GEN sînt salvate în stivă (pînă la 8 nivele) şi rămîn active pînă sînt explicit redefinite.
RESTORE	Se refac opţiunile LIST, COND şi GEN salvate ultima dată (din vîrful stivei).
INCLUDE *(fişier)*	Fişierul de cod sursă specificat este inclus în continuare, în fişierul ce conţine opţiunea. INCLUDE pot fi cuprinse pînă la 4 nivele. Liniile incluse sînt marcate cu (=) în coloana 19. În coloana 18 se afişează nivelul de INCLUDE. Această opţiune este foarte utilă pentru gestiunea unor fişiere de macrodefiniţii.

Opţiunile implicite considerate de ASM80 sînt următoarele :

OBJECT (fişier sursă.OBJ)
NODEBUG
PRINT (fişier sursă.LST)
COND
LIST
SYMBOLS
PAGING
PAGELENGTH (66)
PAGEWIDTH (120)
NOXREF
NOMACROFILE
NOMACRODEBUG
GEN
NOTTY

Toate opţiunile arătate mai sus pot fi specificate în linii de control incluse în programul sursă. Formatul acestor linii este :

$ *listă-opţiuni*

$ trebuie să fie în coloana 1 iar opţiunile separate prin cel puţin un spaţiu.

Liniile ce conţin opţiuni PRIMARE trebuie să apară înaintea primei linii din textul sursă (inclusiv comentarii).

8.2.6. Exemplu de programare în limbaj de asamblare

Programul prezentat în continuare reprezintă un exemplu complet de tratare a întreruperilor la microcalculatorul M118. Exemplul prezentat conţine şi tabela de referinţe pentru programul listat.

```
ASM80 :F1:IN8020.ASM MACROFILE PRINT(:LP:) PAGEWIDTH(72) DEBUG XREF


SFDX-18 8080/8085 MACRO ASSEMBLER  V3.0          IN8020    PAGE     1


  LOC  OBJ          LINE        SOURCE STATEMENT

                       1          NAME     IN8020
                       2 $        TITLE    ('***   MODUL INTRERUPERI PENTRU
                          U.C. M-118   ***')
                       3          PUBLIC   RQENDI,RQELVL,RQDLVL,UDPRI
                       4          PUBLIC   INTINI,INTEI,INTDI,RQSETV
                       5          EXTRN    RQL0EX,RQL1EX,RQL2EX,RQL3EX,RQL4
                         EX
                       6          EXTRN    RQL5EX,RQL6EX,RQL7EX,RQISND,?TIC
                         K
                       7          EXTRN    RQACTV,ENTSLL,?ELR
                       8          CSEG
                       9 ;
                      10 ;  DEFINIREA MASTILOR PENTRU INTRERUPERI
                      11 ;
                      12 IMSK:
  0000 FF             13          DB       0FFH,0FEH,0FCH,0F8H,0F0H
  0001 FE
  0002 FC
  0003 F8
  0004 F0
  0005 E0             14          DB       0E0H,0C0H,080H,0
  0006 C0
  0007 80
  0008 00
                      15 ;
                      16 ;  RUTINA RQENDI
                      17 ;  FUNCTIE: TRANSMITE NON-SPECIFIC EOI LA 8259
                      18 ;  PARAMETRII: NIMIC
                      19 ;  INTOARCE: NIMIC
                      20 ;
                      21 RQENDI:
  0009 3E20           22          MVI      A,EOI
  000B D3FD           23          OUT      OCW2
  000D C9             24          RET
                      25 ;
                      26 ;  ZONA RUTINE PREDEFINITE PENTRU INTRERUPERI
                      27 ;
                      28 INTRUT:
                      29 RUT0:
  000E 010000     E   30          LXI      B,RQL0EX ;BC=ADR. EXCHANGE
                      31 COMRUT:
  0011 E5             32          PUSH     H        ;SALVARE STARE
  0012 D5             33          PUSH     D
  0013 F5             34          PUSH     PSW
  0014 CD0000     E   35          CALL     RQISND   ;TRANSMITE MESAJ INTRERU
                         PERE
  0017 F1             36          POP      PSW      ;REFACERE STARE
  0018 D1             37          POP      D
  0019 E1             38          POP      H
  001A C1             39          POP      B
  001B FB             40          EI
  001C C9             41          RET               ;SFIRSIT INTRERUPERE
                      42 RUT2:
  001D C5             43          PUSH     B
```

```
  LOC  OBJ           LINE          SOURCE STATEMENT

  001E 010000    E     44              LXI      B,RQL2EX ;BC=ADR. EXCHANGE
  0021 C31100    C     45              JMP      COMRUT
                       46 RUT3:
  0024 C5              47              PUSH     B
  0025 010000    E     48              LXI      B,RQL3EX ;BC=ADR. EXCHANGE
  0028 C31100    C     49              JMP      COMRUT
                       50 RUT4:
  002B C5              51              PUSH     B
  002C 010000    E     52              LXI      B,RQL4EX ;BC=ADR. EXCHANGE
  002F C31100    C     53              JMP      COMRUT
                       54 RUT5:
  0032 C5              55              PUSH     B
  0033 010000    E     56              LXI      B,RQL5EX ;BC=ADR. EXCHANGE
  0036 C31100    C     57              JMP      COMRUT
                       58 RUT6:
  0039 C5              59              PUSH     B
  003A 010000    E     60              LXI      B,RQL6EX ;BC=ADR. EXCHANGE
  003D C31100    C     61              JMP      COMRUT
                       62 RUT7:
  0040 C5              63              PUSH     B
  0041 F5              64              PUSH     PSW
  0042 3E0B            65              MVI      A,0BH    ;TRIMITE OCW3 DE CITIRE
                          ISR
  0044 D3FD            66              OUT      OCW3
  0046 DBFD            67              IN       ISR
  0048 E680            68              ANI      80H      ;TEST EXISTA INTR. NIVEL
                           7
  004A C25000    C     69              JNZ      SEND7    ;DACA DA SALT
  004D F1              70              POP      PSW      ;ALTFEL REFACE STAREA
  004E C1              71              POP      B
  004F C9              72              RET
                       73 SEND7:
  0050 F1              74              POP      PSW
  0051 010000    E     75              LXI      B,RQL7EX ;BC=ADR. EXCHANGE
  0054 C31100    C     76              JMP      COMRUT
                       77 ;
                       78 ;   RUTINA UDPRI
                       79 ;   FUNCTIE: ACTUALIZEAZA MASTILE DE INTRERUPERE
                          CF. PRIORITATII
                       80 ;                  TASK-ULUI ACTIV CURENT
                       81 ;   PARAMETRII: NIMIC
                       82 ;   INTOARCE: NIMIC
                       83 ;
                       84 UDPRI:
  0057 010E00          85              LXI      B,0EH    ;BC=DEPLASAMENT PRIORITY
                           IN TD
  005A 2A0000    E     86              LHLD     RQACTV   ;HL=ADRESA TD PENTRU "RU
                          NNING TASK"
  005D 09              87              DAD      B
  005E 7E              88              MOV      A,M      ;A=PRIORITY PENTRU "RUNN
                          ING TASK"
  005F B7              89              ORA      A        ;TEST PRIORITY=0
  0060 C26400    C     90              JNZ      PRNZ     ;DACA NU SALT
  0063 3C              91              INR      A        ;INCREMENT PT. A COMPENS
                          A DECREMENTAREA
```

```
LOC  OBJ           LINE        SOURCE STATEMENT

                     92 PRNZ:
0064 FE81            93          CPI      129       ;TEST PRIORITY>=129
0066 DA6B00     C    94          JC       EINT      ;DACA NU SALT
0069 3E81            95          MVI      A,129     ;TOATE PRI>=129 SINT TRA
                        TATE UNIFORM
                     96 EINT:
006B 3D              97          DCR      A         ;A=NIVELUL DE MASCARE=I
                        6
006C E6F8            98          ANI      0F8H      ;PREGATESTE ÎMPĂRŢIREA L
                        A 16
006E 1F              99          RAR
006F 1F             100          RAR
0070 1F             101          RAR
0071 1F             102          RAR
0072 4F             103          MOV      C,A       ;SALVEAZA IN C NIVELUL D
                        E MASCARE
0073 0600           104          MVI      B,0       ;BC=DEPLASAMENT IN TABEL
                        A DE MASTI
0075 210000     C   105          LXI      H,IMSK
0078 09             106          DAD      B         ;HL=ADRESA MASCA CORESPU
                        NZATOARE NIVELULUI
0079 3A0000     D   107          LDA      STMSK     ;ADAUGA MASCA STATICA (N
                        IVELE FARA RQELVL)
007C B6             108          ORA      M
007D D3EC           109          OUT      IMR       ;TRANSMITE MASCA LA IMR
007F C9             110          RET
                    111 ;
                    112 ;   ZONA CELULELOR CAPCANA. SE COPIAZA IN RAM LA
                        INITIALIZARE
                    113 ;
                    114 TRAP:
0080 C30E00     C   115          JMP      RUT0
0083 00             116          NOP
0084 C30000     E   117          JMP      ?TICK
0087 00             118          NOP
                    119          IRP      LVL,<2,3,4,5,6,7>
                    120          JMP      RUT&LVL
                    121          NOP
                    122          ENDM
0088 C31D00     C   123+         JMP      RUT2
008B 00             124+         NOP
008C C32400     C   125+         JMP      RUT3
008F 00             126+         NOP
0090 C32B00     C   127+         JMP      RUT4
0093 00             128+         NOP
0094 C33200     C   129+         JMP      RUT5
0097 00             130+         NOP
0098 C33900     C   131+         JMP      RUT6
009B 00             132+         NOP
009C C34000     C   133+         JMP      RUT7
009F 00             134+         NOP
0020                135 LGTRAP   EQU      $-TRAP
                    136 ;
                    137 ;    RUTINA INTINI
                    138 ;    FUNCTIE: INITALIZAREA SISTEMULUI DE INTRRUPE
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

                         RI
                    139 ;    PARAMETRII: NIMIC
                    140 ;    INTOARCE: NIMIC
                    141 ;
                    142 INTINI:
  00A0 3EE0         143             MVI     A,0E0H  ;A=MASCA MULTIPLU DE 32
  00A2 212300    D  144             LXI     H,RAMTRP+31 ;HL=ADRESA IN RAM A
                         CELULELOR CAPCANA
  00A5 A5           145             ANA     L          ;CORECTIE LA MULTIPLU DE
                          32
  00A6 6F           146             MOV     L,A
  00A7 220200    D  147             SHLD    ADTRAP  ;SALVEAZA ADRESA CORECTA
                         TA
  00AA 1E20         148             MVI     E,LOTRAP ;E=LUNGIME CELULE CPCAN
                         A
  00AC 018000    C  149             LXI     B,TRAP  ;BC=ADRESA DE INCEPUT CO
                         PIERE
                    150 MOVE:
  00AF 0A           151             LDAX    B          ;IA UN OCTET DIN ROM
  00B0 77           152             MOV     M,A        ;IL PUNE IN RAM
  00B1 03           153             INX     B          ;INC. POINTER ROM
  00B2 23           154             INX     H          ;INC. POINTER RAM
  00B3 1D           155             DCR     E          ;DEC. CONTOR
  00B4 C2AF00    C  156             JNZ     MOVE       ;REIA CICLUL
  00B7 2A0200    D  157             LHLD    ADTRAP
  00BA 7D           158             MOV     A,L        ;A=ADR. LOW A CELULELOR
                         CAPCANA
  00BB F616         159             ORI     16H        ;FORMEAZA ICW1 CU "SINGU
                         R" SI "INTERVAL 4"
  00BD D3FD         160             OUT     ICW1       ;TRIMITE ICW1
  00BF 7C           161             MOV     A,H        ;A=ADR. HIGH A CELULELOR
                          CAPCANA
  00C0 D3FC         162             OUT     ICW2       ;TRIMITE ICW2
  00C2 97           163             SUB     A
  00C3 320100    D  164             STA     MSGINI     ;INITIAL NU EXISTA MESAJ
                          INITIALIZAT
  00C6 3EFF         165             MVI     A,0FFH
  00C8 320000    D  166             STA     STMSK      ;INIT STATIC MASK
  00CB D3FC         167             OUT     IMR        ;INITIAL TOATE NIVELELE
                         SINT MASCATE
  00CD 210000       168             LXI     H,0
  00D0 220000    E  169             SHLD    RQL1EX     ;INITIALIZARE DESCRIPTOR
                          RQL1EX
  00D3 220400    E  170             SHLD    RQL1EX+4
  00D6 210000    E  171             LXI     H,RQL1EX
  00D9 220200    E  172             SHLD    RQL1EX+2
  00DC 220600    E  173             SHLD    RQL1EX+6
  00DF 210000    E  174             LXI     H,?ELR  ;HL=ADRESA CAP DE LISTA
  00E2 E5           175             PUSH    H
  00E3 010000    E  176             LXI     B,RQL1EX ;BC=ADR. OBIECT DE INTR
                         ODUS
  00E6 110800       177             LXI     D,8        ;DE=ADR. CIMP DE LEGARE
  00E9 CD0000    E  178             CALL    ENTSLL ;LEAGA RQL1EX IN LISTA EX
                         CHANGE-URI
  00EC C9           179             RET
```

```
  LOC  OBJ         LINE        SOURCE STATEMENT

                    180 ;
                    181 ;    MASTI SETARE BITI
                    182 ;
                    183 BITSET:
 00ED 01            184          DB       1,2,4,8,10H
 00EE 02
 00EF 04
 00F0 08
 00F1 10
 00F2 20            185          DB       20H,40H,80H
 00F3 40
 00F4 80
                    186 ;
                    187 ;    ADRESELE EXCHANGE-URILOR DE INTRERUPERE
                    188 ;
                    189 INEXAD
                    190          IRP      LVL,<0,1,2,3,4,5,6,7>
                    191          DW       RQL&LVL&EX
                    192          ENDM
 00F5 0000     E    193+         DW       RQL0EX
 00F7 0000     E    194+         DW       RQL1EX
 00F9 0000     E    195+         DW       RQL2EX
 00FB 0000     E    196+         DW       RQL3EX
 00FD 0000     E    197+         DW       RQL4EX
 00FF 0000     E    198+         DW       RQL5EX
 0101 0000     E    199+         DW       RQL6EX
 0103 0000     E    200+         DW       RQL7EX
                    201 ;
                    202 ;    RUTINA INTEI
                    203 ;    FUNCTIE: ACTIVEAZA UN NIVEL DE INTRERUPERE P
                        RIN STERGEREA
                    204 ;             BITULUI CORESPUNZATOR DIN STATIC MA
                        SK
                    205 ;             DACA NU EXISTA MESAJ DE INTRERUPERE
                         FORMAT LA
                    206 ;             EXCHANGE-UL CORESPUNZATOR NIVELULUI
                         FORMEAZA UNUL
                    207 ;    PARAMETRII: C=NR. NIVELULUI DE ACTIVAT
                    208 ;    INTOARCE: NIMIC
                    209 ;
                    210 INTEI:
 0105 0600          211          MVI      B,0      ;BC=NR.NIVEL
 0107 21ED00   C    212          LXI      H,BITSET
 010A 09            213          DAD      B
 010B 5E            214          MOV      E,M      ;E=BIT CORESPUNZATOR NIV
                        ELULUI
 010C 7B            215          MOV      A,E      ;SI IN A
 010D 210100   D    216          LXI      H,MSGIN1
 0110 A6            217          ANA      M        ;TEST MESAJ INITIALIZAT
 0111 C23401   C    218          JNZ      NOINIT   ;DACA DA SALT
 0114 21F500   C    219          LXI      H,INEXAD ;HL=BAZA TABELEI DE ADR
                        ESE EXCHANGE-URI
 0117 09            220          DAD      B
 0118 09            221          DAD      B        ;HL=ADRESA ADRESEI EXCHA
                        NGE
```

```
 LOC  OBJ           LINE         SOURCE STATEMENT

 0119 4E             222          MOV     C,M
 011A 23             223          INX     H
 011B 46             224          MOV     B,M       ;BC=ADR. EXCHANGE CAUTAT
 011C 210A00         225          LXI     H,10      ;HL=ADR. CIMP "LINK" IN
                         MESAJ
 011F 09             226          DAD     B
 0120 75             227          MOV     M,L       ;"LINK" PE EL INSUSI
 0121 7C             228          MOV     A,H
 0122 23             229          INX     H
 0123 77             230          MOV     M,A
 0124 23             231          INX     H         ;HL=ADR. "LENGTH"
 0125 3605           232          MVI     M,5       ;LENGTH=5
 0127 23             233          INX     H
 0128 3600           234          MVI     M,0
 012A 23             235          INX     H         ;HL=ADR. "TYPE"
 012B 3601           236          MVI     M,1       ;TYPE=1
 012D 3A0100    D    237          LDA     MSGINI
 0130 B3             238          ORA     E         ;MARCHEAZA INITIALIZAREA
                          MESAJULUI
 0131 320100    D    239          STA     MSGINI
                     240 NOINIT:
 0134 210000    D    241          LXI     H,STMSK
 0137 7B             242          MOV     A,E       ;STERGE BITUL CORESPUNZA
                         TOR DIN STATIC MASK
 0138 2F             243          CMA
 0139 A6             244          ANA     M
 013A 77             245          MOV     M,A
 013B C35700    C    246          JMP     UDPRI     ;MODIFICA IMR
                     247 ;
                     248 ;    RUTINA RQELVL
                     249 ;    FUNCTIE: ACTIVEAZA UN NIVEL DE INTRERUPERE P
                         RIN
                     250 ;              APELAREA RUTINEI INTEI INTR-O SECVE
                         NTA INDIVIZIBILA.
                     251 ;    PARAMETRII: C=NR. NIVEL
                     252 ;    INTOARCE: NIMIC
                     253 ;
                     254 RQELVL:
 013E F3             255          DI                ;INCEPUT SECVENTA INDIVI
                         ZIBILA
 013F CD0501    C    256          CALL    INTEI     ;ACTIVARE NIVEL
 0142 FB             257          EI                ;SFIRSIT SECVENTA INDIVI
                         ZIBILA
 0143 C9             258          RET
                     259 ;
                     260 ;    RUTINA INTDI
                     261 ;    FUNCTIE: DEZACTIVEAZA UN NIVEL DE INTRERUPER
                         E
                     262 ;              PRIN MODIFICAREA STATIC MASK-ULUI
                     263 ;    PARAMETRII: C=NR. NIVEL
                     264 ;    INTOARCE: NIMIC
                     265 ;
                     266 INTDI:
 0144 0600           267          MVI     B,0       ;BC=NR. NIVEL
 0146 21ED00    C    268          LXI     H,BITSET
```

```
  LOC  OBJ          LINE        SOURCE STATEMENT

  0149 09            269            DAD     B          ;HL=ADR. BIT MASCA PT. N
                         IVELUL RESPECTIV
  014A 7E            270            MOV     A,M        ;ADUCE IN A BITUL MASCA
  014B 210000   D    271            LXI     H,STMSK
  014E B6            272            ORA     M          ;REALIZEAZA MASCAREA
  014F 77            273            MOV     M,A
  0150 C35700   C    274            JMP     UDPRI      ;MODIFICA IMR
                     275 ;
                     276 ;   RUTINA RQDLVL
                     277 ;   FUNCTIE: DEZACTIVEAZA UN NIVEL DE INTRERUPER
                         E PRIN
                     278 ;               APELAREA RUTINEI INTDI INTR-O SECVE
                         NTA INDIVIZIBILA
                     279 ;   PARAMETRII: C=NR. NIVEL
                     280 ;   INTOARCE: NIMIC
                     281 ;
                     282 RQDLVL:
  0153 F3            283            DI                 ;INCEPUT SECVENTA INDIVI
                         ZIBILA
  0154 CD4401   C    284            CALL    INTDI ;DEZACTIVEAZA NIVEL
  0157 FB            285            EI                 ;SFIRSIT SECVENTA INDIVI
                         ZIBILA
  0158 C9            286            RET
                     287 ;
                     288 ;   RUTINA RQSETV
                     289 ;   FUNCTIE: MODIFICA ADRESA DE SALT IN CADRUL U
                         NEI CELULE
                     290 ;               CAPCANA
                     291 ;   PARAMETRII: BC=ADR. NOII RUTINE DE TRATARE A
                          INTRRUPERII
                     292 ;                 E=NR. NIVELULUI PE CARE SE FACE
                          MODIFICAREA
                     293 ;   INTOARCE: NIMIC
                     294 ;
                     295 RQSETV:
  0159 F3            296            DI                 ;INCEPUT SECVENTA INDIVI
                         ZIBILA
  015A 2A0200   D    297            LHLD    ADTRAP     ;HL=ADR. DE INCEPUT A CE
                         LULELOR CAPCANA
  015D 1600          298            MVI     D,0        ;DE=NR. NIVEL
  015F EB            299            XCHG
  0160 29            300            DAD     H
  0161 29            301            DAD     H          ;HL=NR. NIVEL*4 (DEPLASA
                         MENT IN ZONA CAPCANE)
  0162 23            302            INX     H          ;TRECE PESTE COD JMP
  0163 19            303            DAD     D          ;HL=ADRESA LA CARE SE FA
                         CE MODIFICAREA
  0164 71            304            MOV     M,C
  0165 23            305            INX     H
  0166 70            306            MOV     M,B        ;PUNE ADR. NOII RUTINE
  0167 FB            307            EI                 ;SFIRSIT SECVENTA INDIVI
                         ZIBILA
  0168 C9            308            RET
                     309 ;
                     310 ;   CONSTANTE I/E 8259
```

```
SFDX-18 8080/8085 MACRO ASSEMBLER, V3.0          IN8020   PAGE    8
***   MODUL INTRERUPERI PENTRU U.C. M-118   ***

  LOC  OBJ          LINE          SOURCE STATEMENT

                     311 ;
  00FD               312 ICW1     EQU      0FDH
  00FC               313 ICW2     EQU      0FCH
  00FD               314 OCW2     EQU      0FDH
  00FD               315 OCW3     EQU      0FDH
  00FD               316 ISR      EQU      0FDH
  00FC               317 IMR      EQU      0FCH
  0020               318 EOI      EQU      020H
                     319 ;
                     320 ;    ZONA DE DATE A MODULULUI
                     321 ;
                     322          DSEG
  0000               323 STMSK:   DS       1         ;STATIC MASK
  0001               324 MSGINI:  DS       1         ;MARCA PREZENTA MESAJE D
                         E INTRERUPERE
  0002               325 ADTRAP:  DS       2         ;ADRESA ZONEI CELULELOR
                         CAPCANA
  0004               326 RAMTRP:  DS       31+32     ;ZONA CELULE CAPCANA IN
                         RAM
                     327          END


PUBLIC SYMBOLS
INTDI  C 0144    INTEI  C 0105    INTINI C 00A0    RQDLVL C 0153
RQELVL C 013E    RQENDI C 0009    RQSETV C 0159    UDPRI  C 0057

EXTERNAL SYMBOLS
?ELR   E 0000    ?TICK  E 0000    ENTSLL E 0000    RQACTV E 0000
RQISND E 0000    RQL0EX E 0000    RQL1EX E 0000    RQL2EX E 0000
RQL3EX E 0000    RQL4EX E 0000    RQL5EX E 0000    RQL6EX E 0000
RQL7EX E 0000

USER SYMBOLS
?ELR   E 0000    ?TICK  E 0000    ADTRAP D 0002    BITSET C 00ED
COMRUT C 0011    EINT   C 006B    ENTSLL E 0000    EOI    A 0020
ICW1   A 00FD    ICW2   A 00FC    IMR    A 00FC    IMSK   C 0000
INEXAD C 00F5    INTDI  C 0144    INTEI  C 0105    INTINI C 00A0
INTRUT C 000E    ISR    A 00FD    LGTRAP A 0020    MOVE   C 00AF
MSGINI D 0001    NOINIT C 0134    OCW2   A 00FD    OCW3   A 00FD
PRNZ   C 0064    RAMTRP D 0004    RQACTV E 0000    RQDLVL C 0153
RQELVL C 013E    RQENDI C 0009    RQISND E 0000    RQL0EX E 0000
RQL1EX E 0000    RQL2EX E 0000    RQL3EX E 0000    RQL4EX E 0000
RQL5EX E 0000    RQL6EX E 0000    RQL7EX E 0000    RQSETV C 0159
RUT0   C 006E    RUT2   C 001D    RUT3   C 0024    RUT4   C 002B
RUT5   C 0032    RUT6   C 0039    RUT7   C 0040    SEND7  C 005G
STMSK  D 0000    TRAP   C 0080    UDPRI  C 0057

ASSEMBLY COMPLETE,   NO ERRORS
```

```
?ELR       7     174
?TICK      6     117
ADTRAP   147     157     297     325#
BITSET   183#    212     268
COMRUT    31#     45      49      53      57      61
EINT      94      96#
ENTSLL     7     178
EOI       22     318#
ICW1     160     312#
ICW2     162     313#
IMR      109     167     317#
IMSK      12#    105
IN8020     1
INEXAD   189#    219
INTDI      4     266#    284
INTEI      4     210#    256
INTINI     4     142#
INTRUT    28#
ISR       67     316#
LGTRAP   135#    148
MOVE     150#    156
MSGINI   164     216     237     239     324#
NOINIT   218     240#
OCW2      23     314#
OCW3      66     315#
PRNZ      90      92#
RAMTRP   144     326#
RQACTV     7      86
RQDLVL     3     282#
RQELVL     3     254#
RQENDI     3      21#
RQISND     6      35
RQL0EX     5      30     193
RQL1EX     5     169     170     171     172     173     176     194
RQL2EX     5      44     195
RQL3EX     5      48     196
RQL4EX     5      52     197
RQL5EX     6      56     198
RQL6EX     6      60     199
RQL7EX     6      75     200
RQSETV     4     295#
RUT0      29#    115
RUT2      42#    123
RUT3      46#    125
RUT4      50#    127
RUT5      54#    129
RUT6      58#    131
RUT7      62#    133
SENB7     69      73#
STMSK    107     166     241     271     323#
TRAP     114#    135     149
UDPRI      3      84#    246     274

CROSS REFERENCE COMPLETE
```

Capitolul 9

Aplicații ale microcalculatoarelor din familia Felix M18

Microcalculatoarele din familia FELIX M18 au intrat în fabricația de serie din anul 1978, fiind folosite ca sisteme cu caracter universal, în cercetare științifică, proiectare și învățămînt sau ca sisteme dedicate, orientate pe aplicații specifice.

Atît sesiunile de comunicări științifice, în domeniul calculatoarelor, organizate în ultimii ani, în țara noastră, cît și o serie de materiale publicate demonstrează utilizarea microcalculatoarelor FELIX M18-118 într-un mare număr de aplicații, în cele mai diverse domenii.

Întrucît este extrem de greu a se cuprinde într-un singur volum multitudinea acestor aplicații, s-a adoptat punctul de vedere de a se prezenta un număr extrem de restrîns.

Astfel, pe lîngă exemplul de proiectare a unui modul SLAVE, se prezintă : Concentratorul de date CD80, Cuplorul de proces SPOT-80 și sistemul SINEX-82. Atît concentratorul de date, cît și cuplorul SPOT-80 sînt produse în mod curent la Întreprinderea de Calculatoare. Ele constitue produse care s-au exportat. Sistemul SINEX-82 a fost conceput în cadrul ICEMENERG, pentru conducerea proceselor termoenergetice. Realizarea lui s-a bazat pe unități de repertoriu comercial din FELIX M118 și SPOT-80, la care s-au adăugat panouri specializate pentru comunicația cu utilizatorul. Sistemul funcționează sub monitorul de timp real RTX-18, constituind astfel un exemplu concludent, privind flexibilitatea și modularitatea sistemelor de microcalculatoare, atît ca hardware, cît și ca software.

Se prezintă aspecte privind proiectarea și implementarea conducerii cu FELIX M18 a procesului de preparare a minereurilor neferoase. Se descriu utilizarea microcalculatorului M18 în sisteme de culegere a datelor și pontaj automat precum și funcțiile sistemului de introducere și validare a datelor.

În finalul capitolului se prezintă un sistem de dezvoltare pentru studiul structurilor multimicroprocesor bazat pe module URC-M18 ; de asemenea, utilizarea familiei M18 în testarea automată.

9.1. Proiectarea unei interfețe corespunzătoare unui modul SLAVE, cuplat pe magistrala sistemelor din familia FELIX M18

Necesitatea proiectării unor interfețe de tip modul SLAVE, pentru magistrala sistemelor din familia FELIX M18, apare în numeroase cazuri,

cînd utilizatorul intenționează să cupleze la sistem echipamente noi, care nu sînt prevăzute în repertoriul de unități comerciale. Astfel, în laboratoarele Catedrei de calculatoare, din Institutul Politehnic București au fost realizate module SLAVE pentru diferite aplicații :

— modul de achiziție de date analogice, cu 16 intrări cu cîte o bornă la masă sau cu 8 intrări diferențiale și cu 2 ieșiri analogice,

— modul pentru achiziția semnalelor numerice,

— modul pentru achiziția de imagini, cu ajutorul unei camere de luat vederi tip TEHNOTON,

— modul pentru afișarea imaginilor, prelucrate cu microcalculatorul, pe un monitor TV TEHNOTON (imagini de 256×256 pixeli, cu 4 biți/pixel),

— modul pentru comanda unui motor pas cu pas,

— modul de cuplare a magistralei sistemului FELIX M18 la o magistrală de tip HP-IB (standardul IEEE 488),

— module pentru programarea de memorii PROM, REPROM etc.,

— modul cu convertoare N/A, pentru cuplarea unui display cu memorie TEKTRONIX și a unui înregistrator X, Y.

Elementele interfeței

Interfața la magistrală, pentru un modul SLAVE, are în componența sa : decodificatorul de adrese, circuitele de atac pentru magistrală și logica semnalelor de comandă.

Decodificatorul de adrese asigură selecția unor celule de memorie (RAM, ROM) sau porturi de intrare/ieșire, pe baza informației furnizate pe liniile de adrese ale magistralei. Pentru a se asigura o mare flexibilitate, în cadrul acestei logici se utilizează o metodă de decodificare a adresei în două trepte, combinată cu posibilitatea unei selecții suplimentare prin comutatoare. Pornind de la numărul necesar de locații unice (celule de memorie sau porturi de I/E) se stabilesc liniile de adrese inferioare, ale magistralei, folosite în acest scop. Liniile superioare de adrese rămase vor fi decodificate în prima treaptă (primul decodificator) și vor furniza adresa bază. Semnalul corespunzător acesteia va activa decodificatorul celei de-a doua trepte, care va genera semnalele de selecție pentru locațiile unice asociate interfeței. În figura 9.1 se prezintă schema decodificării în două trepte a adreselor de pe magistrală. Biții A4—A8 sînt utilizați pentru a produce semnale de selecție, corespunzătoare bazei alese. În cazul decodificării adreselor de bază, pentru porturi de I/E, se folosește decodificatorul 8205 superior, din prima treaptă (adresele A4—A7). Intrarea inferioară a circuitului 74S32 este conectată la masă prin comutatorul 2. Decodificatorul celei de-a doua trepte, pentru liniile A0—A3, este activat de ieșirea porții 74S32, pentru a furniza semnalele de selecție a locațiilor. În cazul manipulării unor locații de memorie (ca porturi de I/E), treapta întîi va fi formată din două decodificatoare 8205.

După cum s-a văzut în capitolul 2, comanda trebuie să devină activă în 50ns, după stabilirea adreselor. De aceea, logica de decodificare a adreselor trebuie să fie cît mai simplă, pentru a nu introduce întîrzieri.

Circuitele de atac pentru magistrală. În cazul în care interfața recepționează date de la magistrală, vor fi prevăzute circuite tampon, pentru a reduce încărcarea liniilor de date ale magistralei. Dacă interfața trebuie să plaseze date pe magistrală, liniile respective vor dispune de circuite

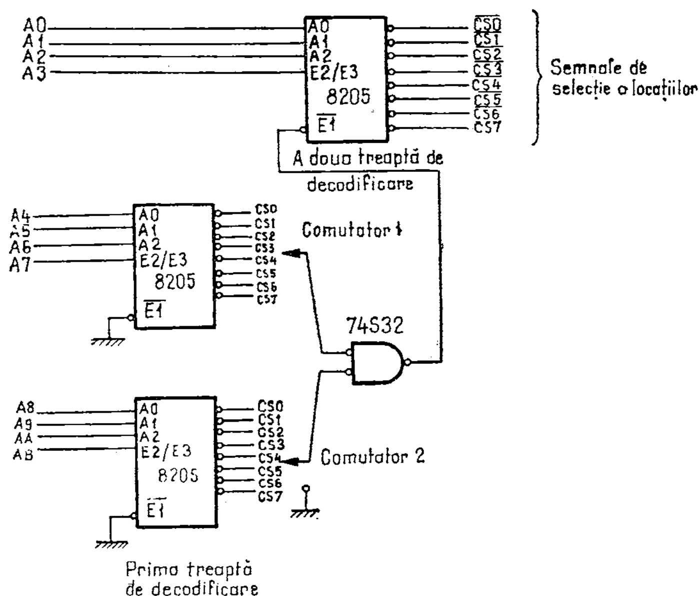


Fig. 9.1. Decodificarea în două trepte a adreselor magistralei.

de atac cu trei stări. Aceste circuite vor fi activate la aplicarea semnalelor IORCL sau MRDCL, simultan cu selecția modulului (activarea adresei bază).

Pentru a asigura operațiile de scriere și citire pe magistrală, de către interfață, se folosesc circuite de atac bidirecționale (8226, 8287). În cazul în care timpul de menținere al informațiilor, furnizate de magistrală, nu este suficient pentru circuitele interfeței, trebuie să se ia măsuri speciale. Acestea constau în activarea permanentă a tamponului receptor și comanda direcției de transfer prin tampoane.

Logica semnalelor de comandă. Această logică cuprinde : circuitele care transmit comenzile de citire/scriere de la magistrală (MRDCL, MWTCL, IORCL, IOWCL) spre destinațiile lor, pe interfață, circuitul pentru generarea semnalului de confirmare a transferului (XACKL) și circuitul care generează semnale de întrerupere, pentru liniile magistralei, afectate în acest scop (RINT0L-RINT7L).

Liniile care primesc semnalele de comandă MRDCL, MWTCL, IORCL și IOWCL trebuie să fie prevăzute cu circuite tampon rapide (Shottky) pentru a nu încărca magistrala și pentru a asigura o imunitate ridicată la zgomot. Aceste semnale sînt validate de semnalul de la decodificatorul adresei bază, în vederea generării comenzilor de citire/scriere în modulul SLAVE.

Semnalul de confirmare a transferului, XACKL, este generat de modulul SLAVE, pentru a informa modulul MASTER în legătură cu citirea datelor transmise de către acesta din urmă sau în legătură cu disponibilitatea pe magistrală a datelor furnizate de modulul SLAVE. Semnalul XACKL este furnizat la linia respectivă a magistralei printr-un circuit cu trei stări, activat atunci cînd modulul SLAVE este adresat și comanda (citire/scriere) este prezentă. Este necesară asigurarea unei anumite flexi-

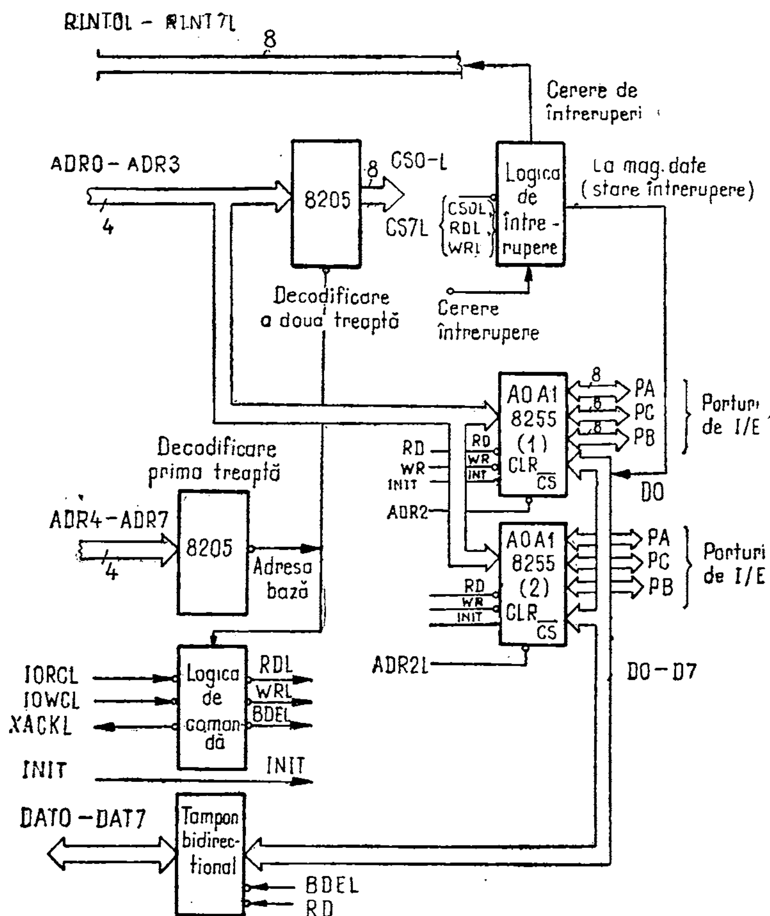


Fig. 9.2. Schema la nivel funcțional a interfeței.

bilități în generarea acestui semnal pentru a putea satisface cerințele UCP-MASTER și ale logicii utilizatorului.

Semnalele asincrone pentru cererile de întrerupere sînt generate de către modulul SLAVE, pe liniile RINT0L-RINT7L, prin circuite cu colectorul deschis, capabile să comande un curent de minimum 16 mA. În sistemele cu întreruperi nevectorizate, cererea de întrerupere se generează forțîndu-se pe magistrală și se memorează într-un bistabil local, care va fi citit de către UCP, printr-o instrucțiune de intrare, cu adresa corespunzătoare. Anularea cererii memorate în acest bistabil se va face printr-o instrucțiune de ieșire.

În cele ce urmează se va prezenta metodologia de proiectare a unui modul SLAVE, echipat cu două interfețe 8255, capabil să genereze un semnal de întrerupere.

Schema la nivel funcțional este dată în figura 9.2. În această schemă se evidențiază : cele două decodificatoare 8205, care formează prima și cea de-a doua treaptă, în vederea selecției, logica de comandă care preia

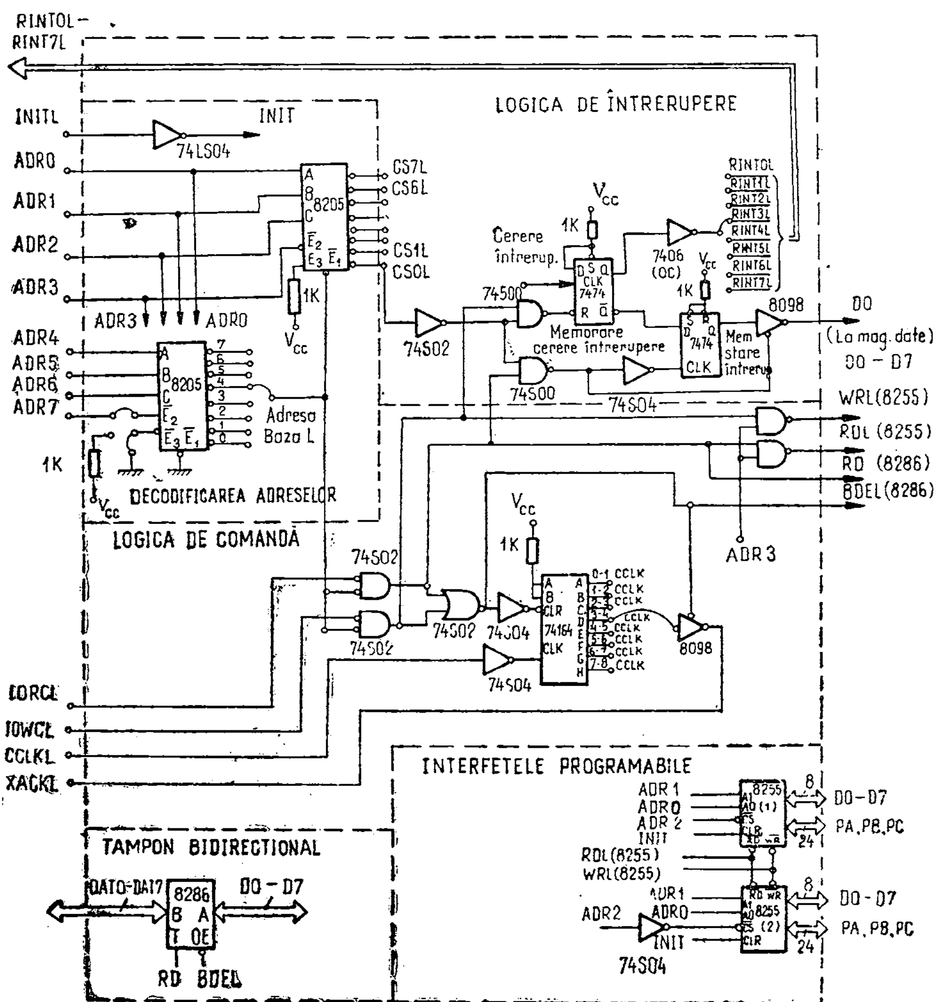


Fig. 9.3. Schema detaliată a interfeței.

de la magistrala sistemului semnalele IORCL, IOWCL, INT7L și generează XACKL, tamponul bidirecțional de date, logica de întrerupere, care la primirea unei cereri de întrerupere generează un semnal RINT3L și interfețele paralele programabile 8255.

Schema detaliată este prezentată în figura 9.3, unde sînt delimitate elementele funcționale amintite mai sus. Față de schema originală [6.] s-au făcut adaptările necesare ținînd seama de specificul semnalelor magistralei sistemului FELIX M18. Astfel, liniile de adrese ADRO-ADR3 nu au mai fost inversate, iar tamponul bidirecțional de date a fost realizat prin circuitul 8286, care nu introduce o inversare a semnalelor.

Decodificarea adreselor se face în două trepte. În prima treaptă se decodifică biții ADR4—ADR7, pentru a furniza adresa bază, care în cazul de față va fi C. Decodificatorul din treapta a doua se va activa atunci

cînd ADR2 este 0. Astfel, adresele porturilor de I/E alocate acestei interfețe, pentru care vor exista semnale de selecție CS0L-CS7L, la al doilea decodificator, sînt C0—C7. Dintre acestea s-a folosit numai adresa C0, pentru ca, în conjuncție cu o instrucțiune de intrare, să se citească starea bistabilului cererii de întrerupere, iar în conjuncție cu o instrucțiune de ieșire să anuleze conținutul acestui bistabil.

În ceea ce privește manipularea porturilor corespunzătoare celor două interfețe paralele programabile, adresele folosite vor acoperi zona C8—CF. Aceasta rezultă din faptul că ADR3 trebuie să fie activ pentru a se genera semnalele WRL și RDL, necesare celor două interfețe. Prima interfață va fi manipulată cu instrucțiuni de I/E avînd adresele C8—CB (adresa CB numai pentru scrierea comenzii). A doua interfață va utiliza restul adreselor : CC—CF (adresa CF numai pentru scrierea comenzii).

Tamponul bidirecțional de date folosește circuitul 8286 cu I/E neinversate. Intrarea T stabilește sensul transferului (T=1, transfer de la A la B ; T=0, transfer de la B la A) și este activată de semnalul RD, prezent la selecția modulului (adresa bază), în asociație cu o instrucțiune de citire. Intrarea CE, activă pe nivel coborît va contribui la generarea comenzilor necesare porților cu trei stări, pentru a le activa corect, în conjuncție cu semnalul aplicat la T. Semnalul BDEL va fi prezent la selecția modulului, în asociație cu instrucțiunile de citire/scriere. Circuitul 8286 va conduce în direcția B la A, atunci cînd nu se execută o operație de I/E, comandată de către magistrală. Acest lucru este necesar pentru a nu se mai comuta sensul de transfer la o comandă de scriere, care s-ar genera după o perioadă inactivă. De asemenea, circuitele 8255 necesită ca datele să rămînă stabile încă 30ns, după ce s-a înlăturat comanda de scriere, condiție care se asigură, dacă sensul transferului nu se modifică, după dispariția semnalului de scriere.

Semnalele de comandă pentru modulul SLAVE sînt generate pe baza semnalelor de comandă de la magistrală IORCL, IOWCL, care sînt validate de semnalul adresei bază. Astfel, logica de comandă asigură comenzile WRL, RDL, pentru interfețele paralele 8255, semnalele de control pentru tamponul de date și semnalul de confirmare XACKL. Acesta din urmă este furnizat cu ajutorul registrului de deplasare 74164, al cărui conținut este anulat de semnalul BDEL, atunci cînd nu se asigură un acces la modulul SLAVE. Toate ieșirile registrului sînt la nivel coborît. În situația în care intrarea CLR va trece pe nivel ridicat, intrările A și B fiind active, registrul le va forța la ieșiri, cu frecvența semnalului CCLKL. Semnalele obținute la ieșirile registrului de deplasare vor apărea după un număr de perioade de tact. Cu ajutorul unui comutator se poate selecta întîrzierea dorită în activarea lui XACKL. În exemplul de față s-a luat un interval de 300—400ns, deoarece CCLKL apare asincron față de activarea comenzii.

Logica de întrerupere folosește un bistabil de tip D pentru a înregistra o cerere asincronă de întrerupere, de la echipamentele externe, cuplate la acest modul SLAVE. Ieșirea acestui bistabil este cuplată, printr-o poartă cu colectorul deschis, la una din liniile RINTiL. Starea acestui bistabil este, de asemenea, transferată într-un al doilea bistabil de tip D, cînd se execută o instrucțiune de citire cu adresa C0. Prin circuitul cu trei stări 8098, bitul de date D0 al magistralei D0—D7 va fi,

de asemenea, poziționat în 1, la efectuarea instrucțiunii IN C0. Acest nou bistabil a fost introdus pentru a reduce posibilitatea unei întreruperi asincrone, atunci cînd are loc citirea stării întreruperii de către modulul MASTER. Anularea conținutului bistabilului cererii de întrerupere se efectuează cu instrucțiunea OUT C0.

În principiu pot exista mai multe surse de întrerupere, conectate pe aceeași linie (RINT3L, în cazul de față). Modulul MASTER poate determina sursa de întrerupere, prin integrarea bistabililor de memorare a stării întreruperilor asociați cu cererile respective.

Funcționarea interfețelor paralele programabile este evidentă avînd în vedere adresele care le-au fost atribuite și modul de operare descris în capitolul 2.

Folosind o tehnică similară, pe baza unui singur circuit 8255, s-a realizat o interfață între magistrala sistemului FELIX M18 și magistrala HP-IB (IEEE-488). Generarea semnalului XACKL a fost asigurată printr-un monostabil, declanșat cu ocazia unei operații de I/E, la modulul selectat. Interfața, relativ simplă ca hardware, constituie o soluție, bazată în special pe software, pentru problema cuplării la sistemele FELIX M18-118, a unor echipamente compatibile cu standardul HP-IB (IEEE-488).

9.2. Multiplexorul concentratorului de date CD80

Multiplexorul (Codul-80.20) este un echipament care se conectează la microcalculatorul FELIX M18, pentru realizarea economică a legării acestuia cu mai multe terminale de tip asincron lucrînd în mod caracter. Economia de hardware se realizează prin multiplexarea în timp a unei interfețe simple, care există între CD80 și FELIX M18. FELIX M18 comandă și controlează activitatea multiplexorului cu ajutorul instrucțiunilor de intrare/ieșire. Unitatea centrală a calculatorului este solicitată pentru fiecare octet care se transferă cu terminalele conectate la multiplexor. Acesta poate fi echipat cu interfețe full-duplex telefonice (RS 232 C) și/sau telegrafice (buclă de curent).

În afara transferului de date, între terminale și M18, multiplexorul mai asigură : vizualizarea stării interfețelor și a activității pe liniile conectate.

9.2.1. Modulele funcționale

80.20 Logica de bază cuprinde următoarele plachete :
— MX-1 logică de interfață cu M18 ;
— MX-2, MX-3 logică de multiplexare ;
— MX-PC panou pentru vizualizare și sincronizare ;

80.21.
— MX-TG interfață telegrafică full-duplex pentru 4 linii ;
80.22.
— MX-TF interfață telefonică full-duplex pentru 8 linii ;
80.23. Sursa telegrafică.

Numărul maxim de linii instalabile, în funcție de viteza maximă de transmisie pe linie, poate fi de :

— 128 linii, 300 biți/s.
— 80 linii, 600 biți/s.
— 40 linii, 1200 biți/s.

Numărul maxim al plachetelor de interfață : 18.

Caracteristici funcționale (programabile pentru grupuri de 4 linii) :

— Viteze de lucru : 50, 75, 100, 110, 150, 200, 300, 600, 1200, biți/s ;
— Formatul caracterelor : 5, 7, 8 biți/caracter ;
— Număr biți STOP : 1 sau 2 ;
— Paritate : pară, impară, inhibată.

Panoul de vizualizare

— Moduri de lucru :
— Modul vizualizare — oferă posibilitatea de a vizualiza activitatea de emisie/recepție și starea interfețelor pentru grupuri de cîte 8 linii ;
— Modul sincronizat — permite urmărirea procesului de emisie/recepție și vizualizarea parametrilor de linie (viteză, format, paritate) pentru fiecare linie ;
— Vizualizarea se face cu ajutorul unor diode LED ;
— Selecția modurilor de funcționare și a grupelor de linii (sau a liniei în modul SINCRONIZAT) se realizează cu ajutorul unor comutatoare basculante.

9.2.2. Caracteristicile funcționale ale multiplexorului

Terminalele asincrone conectate la multiplexor au interfață standard CCITT V24 (ELA RS 232 C) sau telegrafică (cu emițătoare și receptoare în buclă de curent) bipolară.

Modul de lucru al terminalelor este duplex ; terminalele cu interfață telegrafică pot să lucreze în mod simplex.

Corespunzător celor două tipuri de terminale, multiplexorul poate fi echipat cu interfețe asincrone, de tip standard CCITT V24, denumite și telefonice și interfețe telegrafice. Interfețele asincrone au construcție modulară, modulul de interfețe telefonice conține interfețele pentru 8 linii (terminale), iar modulul de interfețe telegrafice conține interfețele pentru 4 linii.

Formatul, viteza de transfer și tipul parității datelor transferate între multiplexor și terminale se pot selecta pe grupe de cîte 4 terminale.

Formatul datelor este determinat de numărul de biți ai cîmpurilor START-STOP. Se pot utiliza formate START-STOP cu 5,7 sau 8 biți pentru caracter, un bit de START și unul sau doi biți de STOP.

Din punct de vedere al numărului maxim de terminale, admise de multiplexor și al vitezelor posibile pentru transferul datelor, multiple-

xorul se realizează în trei variante : (1) pentru terminale lente, (2) pentru terminale cu viteză medie (varianta de bază) și (3) pentru terminale rapide.

Multiplexorul, în varianta pentru terminale lente, admite un număr maxim de 128 terminale ; vitezele posibile, pentru transferul datelor între terminale și multiplexor, sînt următoarele : 50, 75, 100, 150 și 300 baud.

Varianta de bază a multiplexorului admite maximum 80 de terminale, iar vitezele posibile pentru transferul datelor sînt următoarele : 50, 75, 100, 110, 150, 200, 300 și 600 baud.

Varianta multiplexorului pentru terminale rapide admite maximum 40 de terminale, iar vitezele posibile pentru transferul datelor sînt următoarele : 75, 100, 110, 150, 200, 300, 600 și 1200 baud.

Controlul datelor transferate între terminale și multiplexor se face prin paritate pară sau impară. Controlul parității se poate inhiba.

9.2.3. Utilizarea multiplexorului pentru terminale asincrone

Microcalculatorul M18, echipat cu multiplexorul pentru terminale asincrone, se constituie într-o configurație de echipamente pentru concentrarea datelor denumit CD80 (Concentrator de date 80).

Concentratorul de date, CD80, este utilizat, în mod tipic, ca nod de comutare a datelor, în mod caracter, într-o rețea locală de terminale asincrone (figura 9.4).

Conectarea nodului rețelei locale la un calculator central se realizează prin intermediul interfeței seriale sincrone a microcalculatorului. Modul de lucru al interfeței seriale sincrone a microcalculatorului este duplex sau semiduplex, iar vitezele posibile pentru transferul datelor sînt următoarele : 600, 1200, 4800 sau 9600 baud.

Concentrarea datelor are ca scop eliberarea calculatorului central de sarcinile sistematice legate de transferul datelor în mod caracter ; în plus o amplasare corespunzătoare a nodului rețelei locale oferă avantajul reducerii lungimii totale a liniilor de joasă viteză, prin care terminalele se conectează la multiprocesor.

Terminalele situate la distanță se conectează la multiprocesor prin linii individuale telefonice sau telegrafice, comutate sau concesionate.

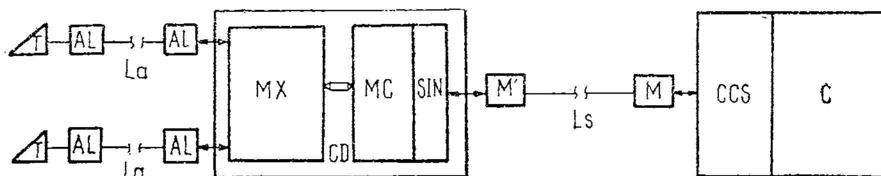


Fig. 9.4. Schemă tipică de utilizare a concentratorului de date CD80 : La — linie asincronă de joasă viteză; Ls — linie sincronă de mare viteză; Mx — multiplexor pentru terminale asincrone ; AL — adaptor de linie pentru linii telegrafice sau modem pentru linii telefonice ; M — modem ; T — terminal ; MC — microcalculator M18; SIN — interfață serială sincronă ; C — calculator central ; CD — concentrator de date ; CCS — canal de comunicație sincron.

Activitatea specifică a concentratorului de date constă în recepţionarea de date în mod caracter, de la terminale, asamblarea unor tampoane pentru fiecare terminal în parte, emisia unor mesaje complete, sincrone şi cu viteză ridicată, către calculatorul central şi funcţia reciprocă de recepţionare de mesaje de la calculatorul central şi emisia acestora, în mod caracter, către terminale.

La nivelul microcalculatorului, utilizatorul poate programa orice modificare a tampoanelor transferate (control sintactic, verificări de cifre de control etc.).

Avînd în vedere existenţa unui microcalculator în configuraţie, concentratorul de date poate fi utilizat şi independent, pentru comutare de mesaje, culegere de date etc.

9.2.4. Structura şi funcţionarea multiplexorului

Multiplexorul pentru terminale asincrone, se compune din următoarele unităţi funcţionale :

— interfaţa cu microcalculatorul M18 ;

— interfeţele asincrone pentru linii telefonice şi telegrafice ;

— logica de bază a multiplexorului ;

— panoul de control.

Din punct de vedere constructiv, multiplexorul se compune din 6 module : MX1, MX2, MX3, MX-TG, MX-TF şi MX-PC.

Modulul MX1 conţine interfaţa cu microcalculatorul, modulele : MX2 şi MX3 conţin logica de bază a multiplexorului, MX-TG este modulul de interfaţă asincronă pentru 4 linii telegrafice, MX-TF este modul de interfaţă asincronă pentru 8 linii telefonice, iar MX-PC este panoul de control.

Multiplexarea datelor

Activitatea multiplexorului pentru terminale asincrone, în ansamblu, se rezumă la realizarea următoarelor funcţiuni :

— recepţionarea datelor transmise serial de către terminale ;

— asamblarea datelor serie recepţionate, în caractere cu format paralel, însoţită de eliminarea biţilor de START şi STOP ;

— memorarea temporară a caracterelor recepţionate ;

— transferul către microcalculator a caracterelor recepţionate însoţite de adresele fizice ale liniilor de la care au fost recepţionate.

De asemenea, multiplexorul execută următoarele funcţiuni reciproce :

— serializarea caracterelor primite de la microcalculator şi generarea biţilor de START şi STOP ;

— emisia serie a semnalelor către terminale.

Deoarece terminalele cuplate la multiplexor lucrează în mod duplex şi în orice moment pot să fie active mai multe terminale, sau chiar toate, multiplexorul trebuie să realizeze simultaneitatea transferului de date între toate terminalele şi microcalculator.

Simultaneitatea transferului datelor între terminale şi microcalculator se realizează, pe de o parte, prin unităţi funcţionale individuale,

pentru fiecare terminal separat, iar pe de altă parte, prin divizarea timpului unităților funcționale utilizate în comun.

Interfețele asincrone pentru linii telefonice și telegrafice au căi separate pentru recepția și emisia datelor serie, pentru fiecare terminal în parte, transferul datelor serie, între interfețele asincrone și logice de bază ale multiplexorului, efectuîndu-se pe căi comune. Datele recepționate serie se transmit logicii de bază, prin linia de intrare date LI (Line Input), utilizată în comun de toate terminalele prin divizarea timpului, iar datele serie emise se transmit interfețelor asincrone, de către logica de bază a multiplexorului, prin linia de ieșire date LO (Line Output) utilizată, de asemenea, în comun de toate terminalele.

Logica de bază a multiplexorului și interfața cu microcalculatorul sînt unități funcționale utilizate în comun de către toate terminalele.

Modul de divizare a timpului pentru unitățile funcționale utilizate în comun, precum și succesiunea în care acestea sînt afectate terminalelor (liniilor asincrone) rezultă din divizarea ciclului multiplexor. Pentru fiecare din cele trei variante de multiplexor ciclul multiplexor are perioadă fixă.

Varianta multiplexor pentru terminale lente are un ciclu de 416 µs, varianta de bază a multiplexorului (pentru terminale cu viteză medie de transfer a datelor) are un ciclu de 208 µs, iar varianta de multiplexor pentru terminale rapide are ciclu de 104 µs.

Divizarea ciclului multiplexor este calitativ aceeași, indiferent de perioada sa (figura 9.5).

Pentru rezolvarea funcțiilor legate de asamblarea datelor serie, primite de la terminale, și serializarea caracterelor, primite de la microcalculator, multiplexorul afectează, din fiecare ciclu multiplexor, un interval de timp de lungime variabilă, funcție de numărul liniilor asincrone instalate la multiplexor și independent de numărul terminalelor. Dacă numărul liniilor asincrone instalate la multiplexor este N, ciclul pentru multiplexarea biților serie se divide în N intervale de timp de cîte 2,4 µs. În aceste intervale de timp unitățile funcționale utilizate în comun sînt afectate succesiv, în ordinea adreselor fizice, liniilor asincrone instalate la multiplexor. În prima jumătate a fiecărui interval de timp, afectat unei linii asincrone, multiplexorul este în starea de recepție (R). Indiferent de

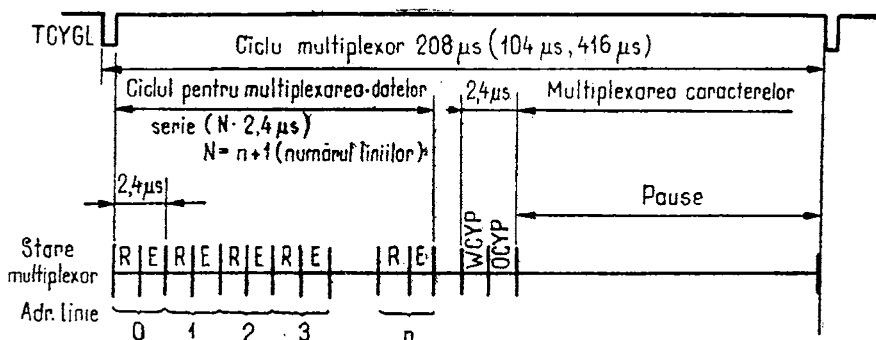


Fig. 9.5. Divizarea ciclului multiplexor.

viteza de lucru a terminalelor în starea de recepție, logica de bază a multiplexorului primește de la interfețele asincrone, prin linia de intrare date, LI, un eșantion din bitul serie, în curs de recepție de la terminalul în cauză. Frecvența eșantioanelor de date prelevate dintr-un caracter serie, în curs de recepție, este egală cu frecvența ciclului multiplexor. Din eșantioanele de date primite, logica de bază a multiplexorului face asamblarea datelor recepționate, în caractere cu format paralel.

Pentru ca asamblarea datelor serie recepționate să fie posibilă, logica de bază a multiplexorului trebuie să primească cel puțin 8 eșantioane din fiecare bit recepționat. Din acest motiv viteza maximă pentru transferul datelor între terminale și multiplexor (biți pe secundă) este de 8 ori mai mică decît frecvența ciclului multiplexor : 1200 baud, pentru ciclu multiplexor de 104 µs, 600 baud pentru ciclu multiplexor de 208 µs și 300 baud pentru ciclu multiplexor de 416 µs.

În cea de a doua jumătate a intervalului de timp de 2,4 µs, afectat unei linii asincrone, multiplexorul este în starea de emisie (E). În această stare, logica de bază a multiplexorului transmite interfețelor asincrone, prin linia de ieșire date, LO, un eșantion din caracterul în curs de serializare, pentru linia asincronă căreia îi este afectată starea de emisie. Deoarece frecvența eșantioanelor de date, transmise unei linii asincrone oarecare, este egală cu frecvența ciclului multiplexor, fiecare bit serie se transmite prin cel puțin 8 eșantioane.

Eșantioanele serie transmise interfețelor asincrone vor fi comutate spre terminalele pentru care sînt destinate.

Pentru rezolvarea funcțiunilor legate de transferul datelor în mod caracter, între multiplexor și microcalculator, multiplexorul afectează un interval de timp de 2,4 µs din fiecare ciclu multiplexor. În prima jumătate a acestui interval de timp, multiplexorul se află în starea de intrare (scriere) date WCYP (Write Cycle) iar în cea de a doua jumătate în starea de ieșire date, OCYP (Output Cycle).

Transferul efectiv al datelor între microcalculator și multiplexor se face în mod asincron în raport cu ciclu multiplexor ; în starea WCYP și OCYP multiplexorul realizează un transfer de date între memoria pentru caractere și tampoanele de intrare-ieșire EDR (Emission Data Register) și RDR (Receive Data Register).

Din starea de ieșire date, OCYP, multiplexorul trece în starea de repaus (PAUSE). Durata acestei stări este variabilă, fiind complementară cu durata ciclului pentru multiplexarea datelor serie.

Ciclul pentru multiplexarea datelor serie.

Ciclul pentru multiplexarea datelor este constituit dintr-o succesiune de stări de recepție și emisie afectate, în ordinea adreselor fizice, liniilor asincrone. Acest ciclu se realizează prin dialogul între logica de bază a multiplexorului și interfețele asincrone, iar controlul dialogului este făcut de comanda comună a logicii de bază.

Emisia constă în serializarea biților caracterului de transmis și încadrarea lor între biții de START și STOP. Caracterul care urmează a fi transmis se încarcă paralel într-un registru de deplasare SR_E. Acesta va primi comenzi de deplasare la dreapta, a căror perioadă va fi egală cu durata unui bit. În figura 9.6 s-au făcut următoarele notații :

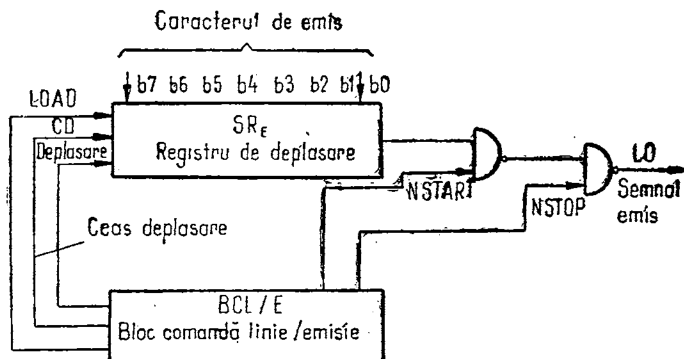


Fig. 9.6. Schema principală a unui emițător pe linia asincronă.

— LO — semnalul emis ;
— SR_E — registrul de deplasare ;
— CD — ceas deplasare.

După momentul încărcării caracterului de emis (comanda LOAD). care este asincronă în raport cu ceasul de deplasare, CD, blocul de comandă, BCL/E, dă comanda de emisie a bitului de START, comandă care este sincronă cu ceasul CD. Se emite apoi bitul b_0 și la ceasul următor se validează deplasarea registrului SR_E (acesta se produce la fiecare tact CD). Comanda de emisie a STOP-ului se dă după emisia ultimului bit, b_7, al caracterului. Vom reține ca elemente esențiale ale unei scheme de serializare : SR_E — registrul de deplasare, care poate fi încărcat paralel și BCL/E — logica de generare a comenzilor.

Recepția constă în deserializarea biților primiți pe linia asincronă. Procesul de deserializare începe după detecția bitului de START și se încheie după ce s-au numărat atîția biți, cîți trebuie să aibă caracterul. La sfîrșitul deserializării se va verifica dacă bitul de STOP este prezent. Caracterul deserializat va fi transferat într-un registru tampon. În figura 9.7 au fost făcute următoarele notații :

— LI — intrare serială — aici apare caracterul ce trebuie deserializat.
— SR_R — registru de deplasare.
— TREC — tact recepție, pentru a detecta bitul de START, linia va fi „examinată“ cu un tact a cărui perioadă este 1/8 din durata unui bit, astfel, va exista o eroare de detecție a START-ului de maximum 12,5%.
— CDS — ceas de deplasare.
— BUFF — registru tampon în care se păstrează caracterul deserializat, acesta se încarcă paralel în registrul SR_R atunci cînd deserializarea este încheiată.
— BCL/R — examinează cu tactul TREC linia LI. Cînd LI trece în O se consideră că s-a detectat START-ul. Se numără 4 tacturi TREC și se dă o comandă CDS. Bitul de START se încarcă în SF_R O. Din mijlocul bitului de START se numără apoi cîte 8 tacturi TREC și se dă comanda de deplasare CDS. În felul acesta, ca valoare a bitului, se ia în considerație valoarea din mijlocul bitu-

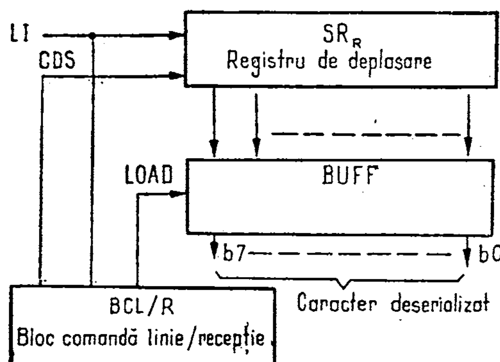


Fig. 9.7. Schema principală a unui receptor pe linia asincronă

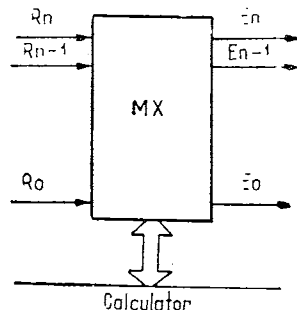


Fig. 9.8. Multiplexorul ca element de legătură între liniile asincrone și calculator.

lui, care are probabilitatea cea mai mare să fie corectă. Atunci cînd toți biții caracterului se află în SR_R se dă comanda de transfer paralel a conținutului acestuia în registrul BUFF.

Multiplexarea în timp a liniilor asincrone (fig. 9.8).

Schema bloc pentru emisie/recepție pe o linie este dată în figura 9.9. Cînd numărul liniilor crește, costul total al unor astfel de blocuri de comandă și costul interfațării lor cu calculatorul este ridicat. Scopul multiplexării liniilor asincrone este de a reduce costul echipamentului. Multiplexarea nu se justifică decît dacă numărul liniilor este de ordinul zecilor.

Nivelul 1 de multiplexare este nivelul de multiplexare în timp al interfeței cu calculatorul.

În figura 9.10 au fost efectuate următoarele notații :

— BCL_0, BCL_i, BCL_n — blocurile de comandă pentru liniile L_0, L_i, L_n ;

— BCMX — blocul de comandă al multiplexorului ;

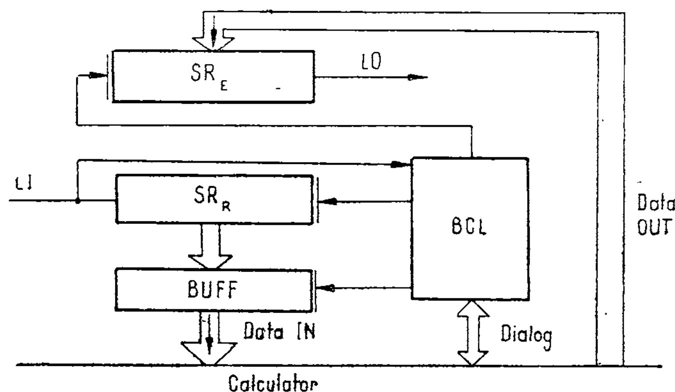


Fig. 9.9. Schema principală pentru emisie/recepție pe o linie asincronă.

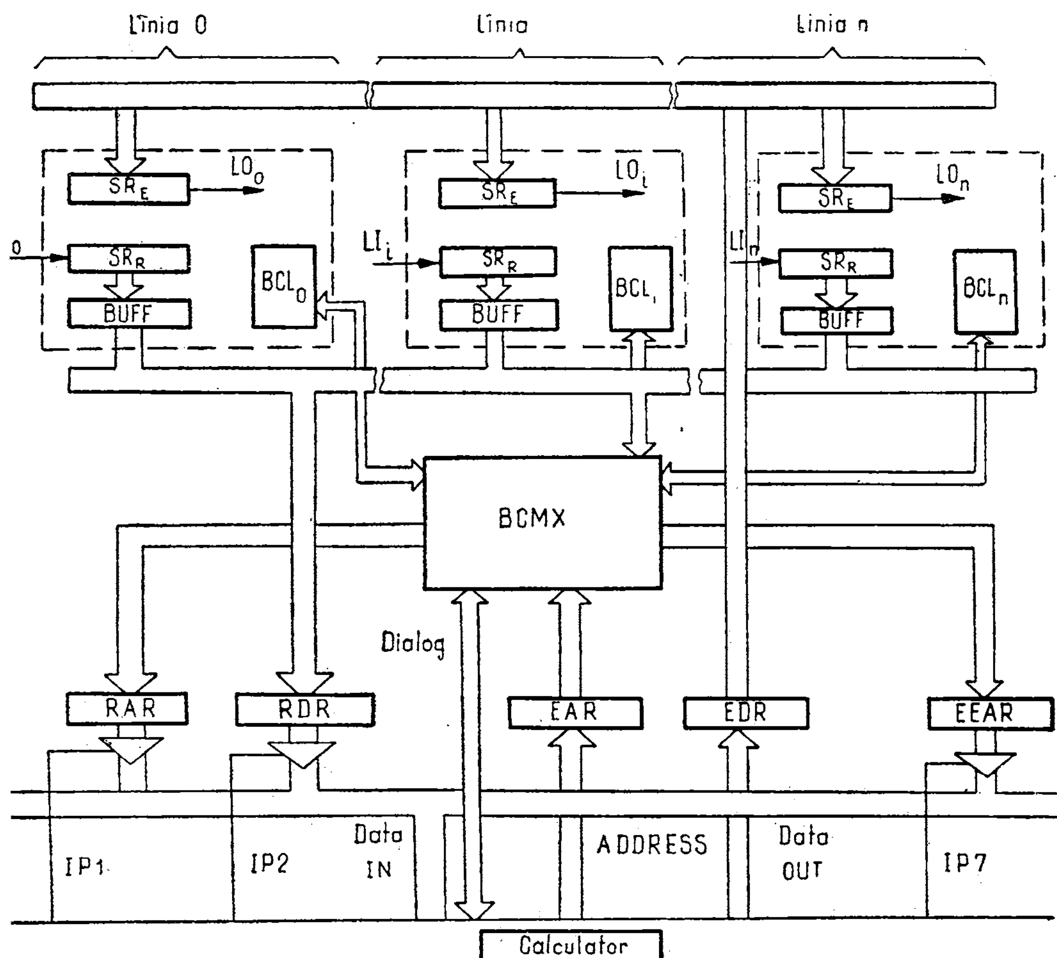


Fig. 9.10. Schema bloc de multiplexor care pune în evidență multiplexarea în timp a interfeței cu calculatorul (multiplexor de nivel 1).

— RAR (recepție) — adresa unei linii pe care s-a recepționat un caracter ;

— RDR (recepție) — caracterul recepționat pe linia a cărei adresă se află în RAR ;

— EAR (emisie) — adresa unei linii pe care se va emite un caracter ;

— EDR (emisie) — caracterul ce trebuie emis pe linia a cărei adresă se află în EAR ;

— EEAR (emisie) — adresa unei linii pe care s-a terminat de emis un caracter ;

— IP1, IP2, IP7 — validează transferul pe magistrală de date al conținutului registrelor RAR, RDR și EEAR.

Emisia. Calculatorul plasează în tamponul de intrare (EAR+EDR) o adresă de linie și un caracter ce trebuie emis pe acea linie. BCMX intră

în dialog cu blocul de comandă al liniei Li, a cărei adresă se află în EAR, iar acesta generează comenzile necesare transferului conținutului registrului EDR în registrul SR_E, asociat liniei Li.

BCLi va genera în continuare toate comenzile necesare în emisie (emisia bitului de START, comenzi de deplasare, emisia bitului de STOP). După emisia bitului de STOP, BCLi intră în dialog cu BCMX, semnalîndu-se faptul că emisia s-a terminat. BCMX va plasa în registrul EEAR adresa liniei Li și semnalează la calculator un sfîrșit de emisie. Calculatorul preia conținutul registrului EEAR (cu comanda IP7) și determină adresa liniei Li, pe care a fost emis caracterul ce l-a trimis într-o secvență anterioară și deci mai poate emite un nou caracter.

În cazul în care pe o linie se termină de emis un caracter și registrul EEAR nu este liber (conținutul său nu a fost preluat de calculator), BCL al acelei linii va rămîne într-o stare de așteptare pentru eliberarea acestui registru.

Recepția. Se consideră că pe o linie Li s-a încheiat deserializarea unui caracter. Acesta va fi transferat în registrul tampon asociat liniei (registrul Buff), iar BCLi va intra în dialog cu BCMX, semnalîndu-i recepția încheiată. BCMX determină adresa liniei și o plasează în registrul RAR, iar caracterul din registrul Buff va fi transferat în RDR. Calculatorul va fi înștiințat că în tamponul de ieșire (RAR+RDR) se află date ce trebuiesc prelucrate.

Dacă tamponul de ieșire este ocupat, BCLi al liniei Li rămîne în starea de așteptare a eliberării acestuia. În timpul acestei stări de așteptare, în registrul SR_R se poate deserializa un nou caracter. Dacă deserializarea se termină și conținutul registrului BUFF nu a fost preluat, BCLi va semnaliza eroare de ritm.

Schema din figura 9.10 evidențiază folosirea registrelor RAR, RDR, EAR, EDR, EEAR pentru interfațarea a n blocuri de emisie/recepție (a căror schemă simplificată a fost dată în figura 9.9).

Următorul nivel de multiplexare, realizează multiplexarea în timp a registrelor SR_{Ei}, SR_{Ri}, $BUFF_i$ și a blocului de comandă BCL_i ($i=0, 1, 2, \ldots, n$).

Se consideră că toate liniile lucrează cu aceeași viteză și anume cu viteza de 1200 biți/s. Durata unui bit la această viteză este aproximativ 832 µ. Perioada tactului de recepție TREC va fi $832\ \mu s/8 \simeq 104\ \mu s$.

BCMX are o bază de timp care este folosită de către toate BCLi. Prin aceasta funcționarea tuturor blocurilor BCLi este sincronizată, ceea ce înseamnă că, atît în emisie, cît și în recepție, comenzile date registrelor de deplasare SR_{Ei} și SR_{Ri} sînt sincrone ($i=0, 1, 2, \ldots, n$).

Se presupune că toate liniile sînt în recepție așteptînd apariția bitului de START. BCL_i ($i=0, 1, 2, \ldots, n$) examinează liniile LI_i pentru a detecta START-ul. BCL_i fiind sincronizate înseamnă că toate LI_i ($i=0, 1, 2, \ldots, n$) vor fi examinate sincron cu un tact, TREC, a cărui perioadă este 104 µs. Se observă că între 2 tacturi TREC nu are importanță starea liniilor LI_i ($i=0, 1, 2, \ldots, n$). Această observație stă la baza multiplexării pe nivelul 2. La momentele date de TREC, se vor lua eșantioane de pe toate liniile LI_i ($i=0, 1, 2, \ldots, n$). Aceste eșantioane vor fi analizate pe rînd între două tacturi. Din caracterul secvențial al acestei analize re-

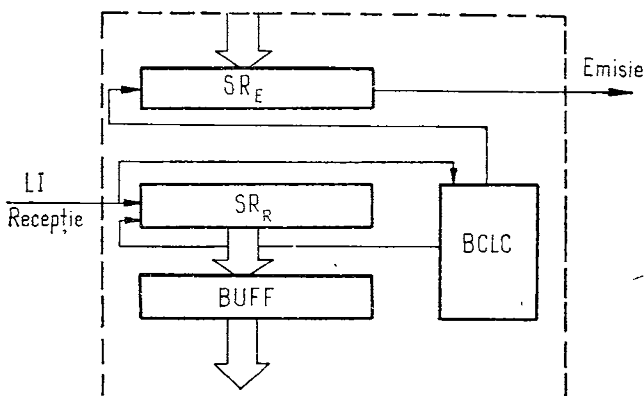


Fig. 9.11. Element de control pentru o linie asincronă.

zultă posibilitatea multiplexării în timp a registrelor de deplasare și a blocurilor de comandă.

Se definește ciclul de analiză, intervalul între două tacturi de eșantionare. În cursul unui ciclu de analiză un element de control de forma celui din figura 9.11 va fi comutat pe rînd pentru analiza fiecărei linii.

Acest mod de lucru presupune existența unor elemente de memorare, care vor păstra conținutul registrelor SR_E, SR_R și BUFF și al stării blocului de comandă, între două analize ale aceleiași linii, făcute în două tacturi diferite.

Cînd în cursul unui ciclu de analiză se examinează o linie Li oarecare, se procedează astfel :

— Se încarcă din memorie starea din tactul anterior a registrelor SR_E, SR_R, BUFF și starea blocului de comandă. Această fază o putem numi de inițializare.

— Blocul de comandă inițializat va da comenzi corespunzătoare, pentru modificarea conținutului registrelor și evoluția într-o stare următoare a proceselor de serializare sau/și deserializare. Blocul de comandă conține elemente cu memorie (numărătoare pentru serializare și deserializare etc.) și de acces.

— Conținutul registrelor SR_E, SR_R și BUFF, precum și noua stare a blocului de comandă sînt rememorate.

Pentru realizarea eșantionării liniilor LI_i se va folosi un registru de deplasare, care are posibilitatea de a fi încărcat paralel.

În continuare se pot face următoarele precizări : referitor la fig. 9.12.

— SDIR se încarcă paralel cu eșantioane de pe liniile Li (i=0, 1, 2, . . . , n).
— În cursul ciclului de analiză, la sfîrșitul analizei fiecărei linii SDIR primește o comandă de deplasare.
— Ieșirea LI va reprezenta la momentul analizei Li eșantionul luat de pe Li.
— După ce s-a examinat linia Li, LO are valoarea eșantionului de bit, care va trebui emis pe Li. Această valoare se încarcă în registrul SDIR.

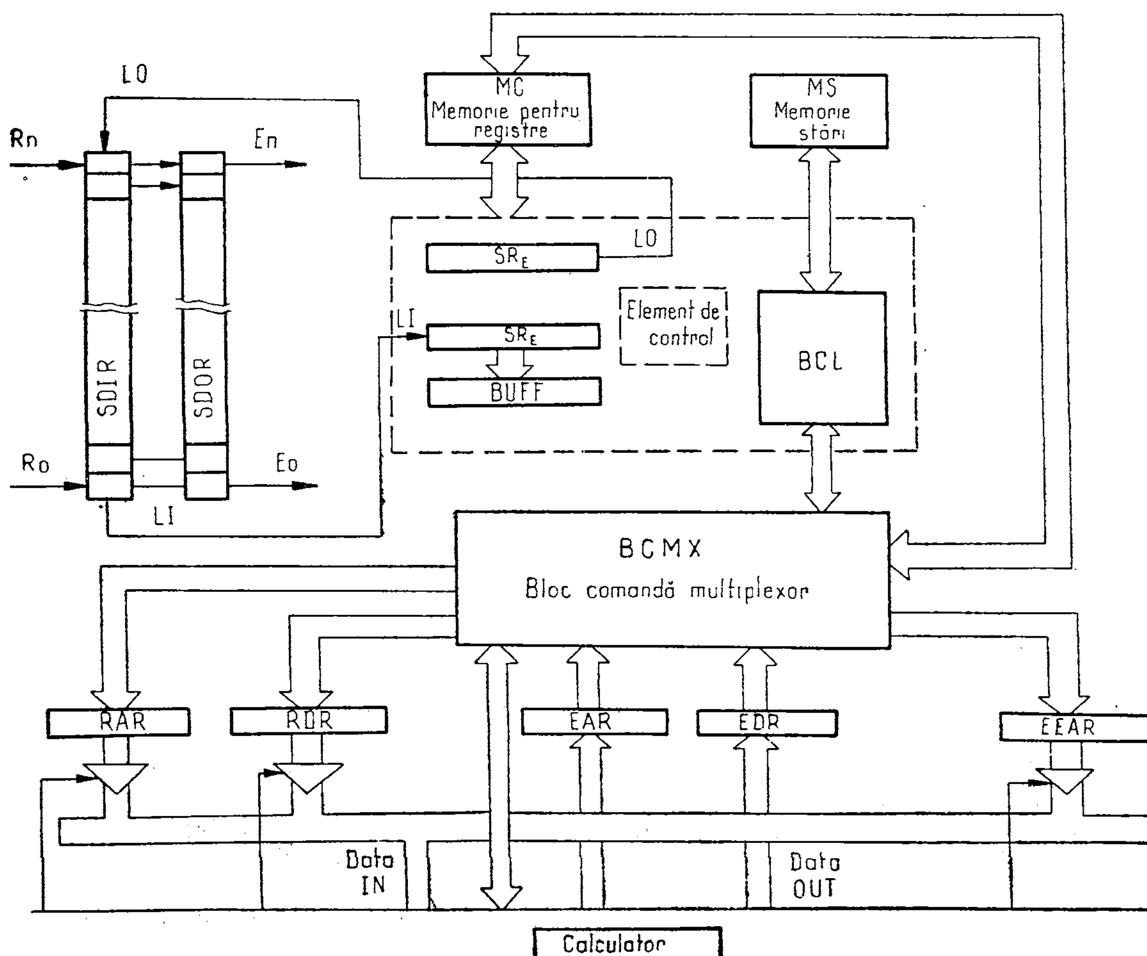


Fig. 9.12. Schema bloc de multiplexor care pune în evidență multiplexarea în timp a interfeței cu calculatorul (nivel 1) și a registrelor de manipulare a caracterelor și a blocurilor de comandă (nivelul 2).

— După ce s-a examinat linia Ln, în poziția 0 a registrului SDIR va fi prezent eșantionul ce trebuie emis pe linia Lo, în poziția 1 eșantionul ce trebuie emis pe linia L1 ș.a.m.d.

— SDOR se încarcă paralel cu eșantionul registrului SDIR. Ieșirile paralele din SDOR atacă emițătoarele de linii. Comanda de încărcare paralelă este aceeași cu a registrului SDIR.

Registrele de interfață cu calculatorul din figura 9.12 au aceeași utilizare ca în schema prezentată în figura 9.10.

— MC constituie memoria în care se păstrează conținutul registrelor SR_E, SR_R și BUFF. Lungimea cuvîntului este egală cu numărul de biți ai registrelor la care se adaugă doi biți funcționali. Pentru fiecare linie sînt rezervate trei cuvinte.

— MS memoria în care se păstrează starea blocului de comandă. Lungimea cuvîntului este determinată de numărul de biți necesari pentru codificarea tuturor stărilor unui bloc BCL. Pentru fiecare linie există un cuvînt.

Schema din figura 9.12 se obține plecînd de la cea din figura 9.10. Considerînd multiplexorul ca în figura 9.10, ambele scheme trebuie să se comporte identic.

În figura 9.10 pentru fiecare linie există un element de control ca cel din figura 9.11. În figura 9.12 există un singur element de control ca cel din figura 9.11 și memorie unde, la locații specifice fiecărei linii, se păstrează imaginea acestui element de control.

Această „complicație“ apărută în schema din figura 9.11 se justifică prin aceea că, costul memoriei pentru n linii, la care se adaugă costul elementului de control comun și al logicii de comutare este mai redus decît costul a n elemente de control, atunci cînd n este de ordinul zecilor.

Starea elementelor de control, la schema din figura 9.10, se poate schimba la fiecare tact TREC, simultan pentru toate liniile. Starea memoriei la schema din figura 9.12 se actualizează între două tacturi, secvențial, astfel încît înaintea unui tact $TREC_{n+1}$, memoria este imaginea fidelă a elementelor de control al liniilor, actualizată conform situației din momentul $TREC_n$. În continuare noțiunile de „examinare a liniei Li“ și „actualizare a elementului de control al liniei Li“ vor fi echivalente.

Actualizarea memoriei se face în cursul ciclului de analiză, definit anterior ca intervalul dintre două tacturi de eșantionare. Intrările de recepție ale liniilor (li(i=0, 1, . . . , n) se eșantionează la intervale de timp discrete, cu un tact TREC a cărui perioadă este $1/8$ din durata unui bit.

Ieșirile de emisie pe liniile Li se schimbă la intervale de timp discrete date, de către același tact TREC. De aici rezultă că un bit emis va fi format din 8 eșantioane. Eșantioanele sînt generate în cursul ciclului de analiză și sînt emise la tactul TREC. Actualizarea elementului de control al liniei Li se face în cursul unui ciclu de analiză după cum urmează :

— se face inițializarea elementului de control comun cu conținutul memorie specifice liniei Li ;

— eșantionul de recepție, luat de pe linia Li, se află la ieșirea LI din registrul SDIR ;

— se actualizează starea registrelor și a blocului de comandă ;

— eșantionul ce trebuie emis se află la intrarea serială a registrului SDIR (intrarea este notată cu LO) ;

— se memorează noua stare în memorie, la locațiile afectate liniei Li ;

— se dă comandă pentru deplasarea registrului SDIR.

La sfîrșitul ciclului de analiză, memoria este actualizată, iar registrul SDIR conține în poziții corespunzătoare fiecărei linii eșantioanele ce trebuiesc emise.

Un nou tact TREC comandă transferul eșantioanelor de emis din SDIR și SDOR și încărcarea paralelă a registrului ADIR cu starea liniilor în recepție.

După terminarea analizei (actualizării) liniei Ln, pînă la un nou tact de eșantionare va exista un interval de timp. În acest interval se va desfășura dialogul cu BCMX pentru încărcarea locațiilor de memorie, unde

se află SRE_i, și pentru transferul locațiilor $BUFF_i$ către tamponul de ieșire. Deci preluarea conținutului tamponului de intrare și încărcarea tamponului de ieșire se fac în fiecare ciclu de analiză.

Dacă, atunci cînd se actualizează elementul de control al unei linii — (în cursul unui ciclu de analiză dintre două momente de eșantionare), recepția și emisia vor fi examinate separat, se vor înlocui registrele SR_E, SR_R și BUFF cu un singur registru SR.

În examinarea recepției pentru linia Li, se va aduce în faza de inițializare conținutul locației de memorie corespunzătoare cu SR_{Ri} și se va încărca în SR. După actualizare se rememorează în aceeași locație. Dacă s-a ajuns la sfîrșitul deserializării, conținutul registrului SR, în loc să fie transferat în BUFF și din BUFF în memorie, este transferat direct în memorie la locația corespunzătoare registrului $BUFF_i$.

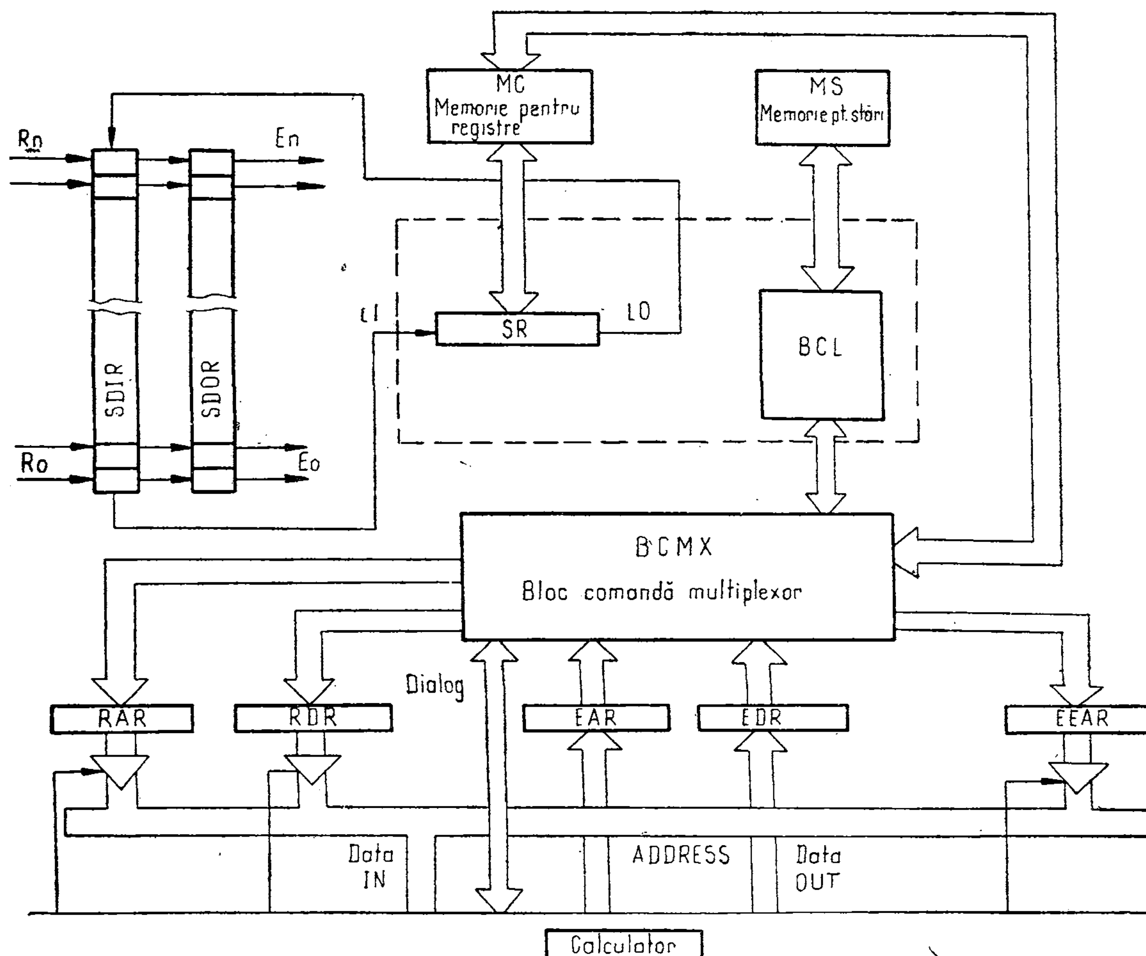


Fig. 9.13. Schema bloc principală a multiplexorului 80.20.

În examinarea emisiei același registru SR va fi încărcat cu conținutul locației de memorie corespunzătoare registrului SR_{Ei}. Apoi se actualizează și rememorează.

Schema din figura 9.13 în care se evidențiază substituirea registrelor SR_E, SR_R și BUFF cu un singur registru SR, este schema bloc principală a multiplexorului 80.20.

9.3. Microcalculatoarele M18, M118 în conducerea proceselor industriale. Cuplorul de proces SPOT80 și exemplificări pentru electroliză și galvanizare

9.3.1. Generalități

În acest paragraf se descriu, din punct de vedere constructiv, caracteristicile calculatoarelor folosite pentru conducerea proceselor, cu exemplificări din sistemul SPOT 80, pentru a da posibilitatea utilizatorilor să găsească soluții, în vederea aplicării lui.

Privind procesul ca fiind compus din trei fluxuri : fluxul material, fluxul energetic și fluxul informațional, scopul calculatorului de proces este de a recepționa datele fluxului informațional și prin algoritmi realizați pe baza tehnologiilor, să mențină fluxul material în parametri stabiliți, iar fluxul energetic în limitele economice.

Caracteristicile principale ale calculatorului de proces sînt : capacitatea de a colecta și transmite cantități mari de informație, de a analiza informațiile culese, a executa operațiile de calcul, conform algoritmilor, și de a emite comenzi către proces.

Pentru a fi asamblate în mod economic, calculatoarele de proces se realizează sub formă de module. Figura 9.14 reprezintă schema bloc a unui astfel de calculator, format din procesorul central și modulele de intrare/ieșire.

Procesorul central este compus din unitatea centrală, memoria fixă (ROM) pentru programe, memoria cu acces aleatoriu (RAM) pentru înregistrarea datelor, sistemul de priorități ale întreruperilor, ceasul de timp real, sistemul de conectare cu un alt calculator, cu o consolă, pentru comunicarea evenimentelor și primirea de comenzi manuale.

Modulele de intrare/ieșire asigură comunicarea cu procesul, conversia numerică a semnalelor emise de traductoarele de măsurare, emiterea de semnale de execuție și semnalizare.

În afara cuplării directe la proces, calculatorul trebuie să posede posibilitatea de comunicare cu operatorul și cu alte calculatoare cu care se află conectat în rețea. Aceste moduri de comunicare se realizează în special prin cuplare serială directă, prin bucla de curent izolată, pentru distanțe pînă la 5 km, și prin MODEM, pentru cuplare prin linii telefonice, și prin cuplare paralelă.

Descrierea procesorului central face obiectul tuturor lucrărilor despre calculatoarele numerice, de aceea nu va fi reluată în prezentul paragraf.

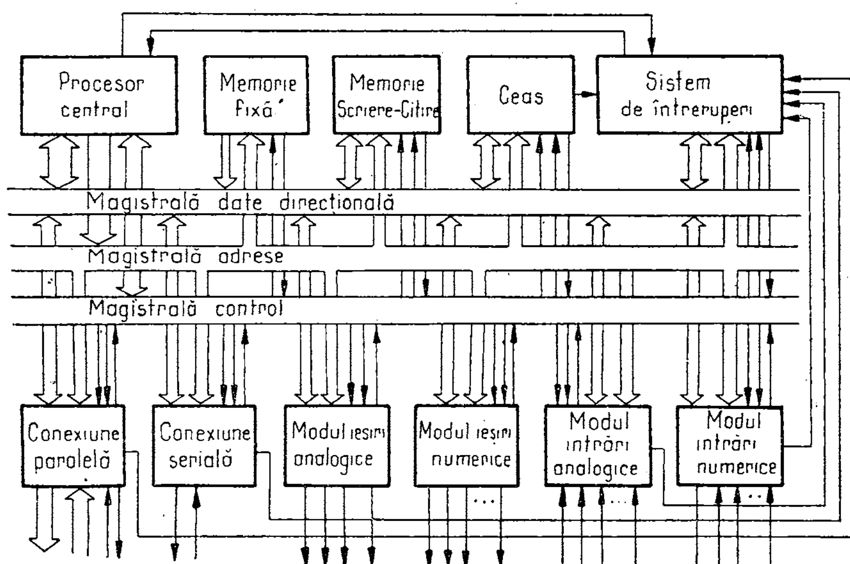


Fig. 9.14. Schema bloc a calculatorului de proces.

Totuși se dau unele explicații asupra sistemului de întreruperi și a ceasului de timp real. În principiu sistemul de întreruperi asigură procesorului central posibilitatea de a face față unor situații urgente, de a comunica cu mai multe module, practic în același timp fără a utiliza metode de așteptare. Sistemul de întreruperi asigură o ierarhizare pe mai multe niveluri, întreruperile cu prioritate superioară pot întrerupe programe cu prioritate inferioare.

Ceasul de timp real asigură o succesiune în timp a executării operațiilor de explorare a mărimilor de intrare și de generare a comenzilor. Evenimentele la care calculatoarele de proces trebuie să răspundă apar la momente și în ordine neprevizibile. Calculatorul înregistrează succesiunea evenimentelor de la intrare, calculează valorile mărimilor de ieșire, care să compenseze abaterile de la starea nominală, și emite comenzile funcție de momentul la care au apărut și însemnătatea lor în proces. Sistemul SPOT 80 are ceasul de timp real construit pe baza circuitului integrat 8253, care conține trei numărătoare de 16 biți, din care două sînt folosite pentru comanda vitezei de emisie a elementelor conectate serial și unul pentru ceas.

9.3.2. Modulele de intrare/ieșire

Comunicarea cu procesul este realizată prin canalele de intrare/ieșire, care conferă de fapt denumirea de calculator de proces unui calculator universal. În figura 9.14 se disting următoarele tipuri de module de intrare/ieșire.

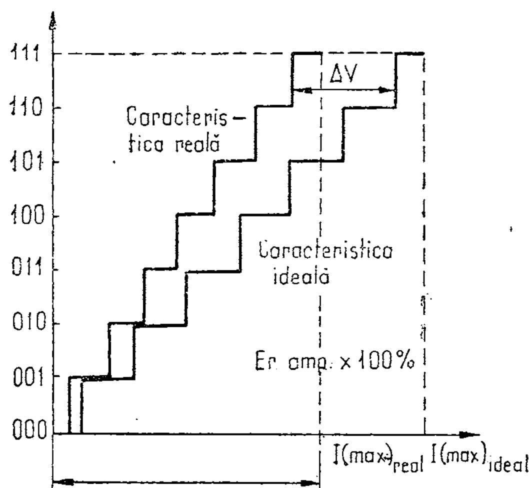


Fig. 9.15. Eroarea de amplificare.

Modulele de intrări analogice sînt modulele de conversie a semnalelor de intrare : tensiuni sau curenți, în cuvinte numerice interpretabile de calculator.

Modulele de ieșiri analogice convertesc cuvintele numerice, rezultate din calculele efectuate de procesorul central, în tensiuni sau curenți ce reprezintă mărimi de comandă pentru elemente de execuție sau mărimi de referință pentru bucle de reglare convenționale.

Modulele de intrări numerice recepționează semnale de tip niveluri de tensiune cu două stări sau supraveghează contacte izolate.

Modulele de ieșiri numerice emit semnale cu două niveluri de tensiune sau comandă ieșirea pe contacte de releu.

Funcționarea modulelor de intrare/ieșire, conversia analog-numerică și numeric-analogică se definesc prin intermediul unor termeni a căror semnificație se dă în cele ce urmează.

Scala sau domeniul maxim de variație a mărimilor de intrare/ieșire este intervalul în care poate varia mărimea analogică, pentru coduri de ieșire/intrare corecte.

Eroarea de amplificare este diferența între valoarea măsurată și valoarea teoretică obținută la ieșirea unui modul pentru o anumită intrare ; ea se exprimă în procente față de domeniul maxim (figura 9.15).

Bit de semnificație maximă este coeficientul termenului cu pondere maximă în reprezentarea unei mărimi, ca suma de puteri ale lui 2 (în formula de mai jos a_n). În cursul lucrării se va utiliza simbolul întîlnit în majoritatea cataloagelor : MSB.

$$V = a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_0 2^0$$

Bit de semnificație minimă este coeficientul termenului cu pondere minimă a_0, în reprezentarea unei mărimi ca sumă de puteri ale lui 2. Este simbolizat în cataloage prin prescurtarea LSB.

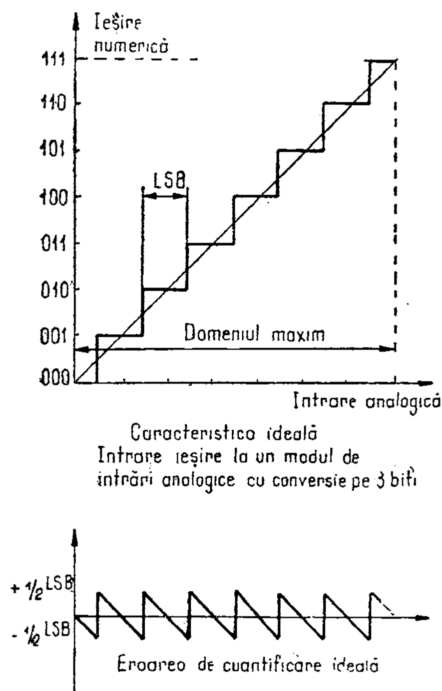


Fig. 9.16. Eroarea de cuantificare ideală.

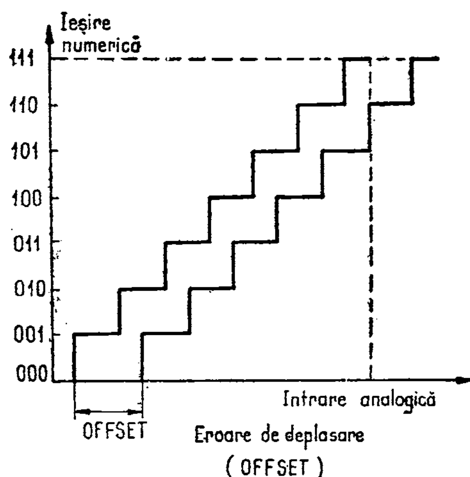


Fig. 9.17. Eroare de deplasare.

Rezoluţia este parametrul ce caracterizează numărul de stări distincte deosebite, la ieşirea unui modul de intrări analogice. De obicei rezoluţia se exprimă în număr de biţi. Pentru modulele de ieşiri analogice, rezoluţia exprimă valoarea variaţiei standard minime a mărimii analogice de ieşire, care corespunde unei succesiuni de două coduri consecutive la intrare, fiind $^1/_2$ din valoarea domeniului de ieşire maxim.

Eroarea de cuantificare se consideră treapta de incertitudine pentru două coduri consecutive (figura 9.16). Pentru un modul cu rezoluţia de N biţi întregul domeniu este divizat în două intervale discrete. Incertitudinea cu care este codificată o valoare în cazul ideal este de $\pm ^1/_2$ LSB.

Eroarea de deplasare (figura 9.17) a caracteristicii de transfer (offset) este mărimea care apare la ieşirea unui modul de ieşiri analogice, cînd la intrarea sa se aplică codul corespunzător valorii zero. Pentru modulele de intrări analogice această eroare este valoarea intrării analogice pentru care se obţine la ieşire valoarea numerică zero.

Eroarea de linearitate (figura 9.18) sau mai scurt linearitatea este diferenţa între valoarea mărimii de ieşire şi valoarea corespunzătoare de pe caracteristica ideală.

Eroarea de linearitate diferenţială (figura 9.19) este diferenţa între treapta de un bit într-un anumit punct şi bitul de semnificaţie minimă LSB.

Comportarea monotonă (figura 9.20) este caracteristica unui modul de intrare sau ieşire analogică de a menţine panta semnalului la ieşire

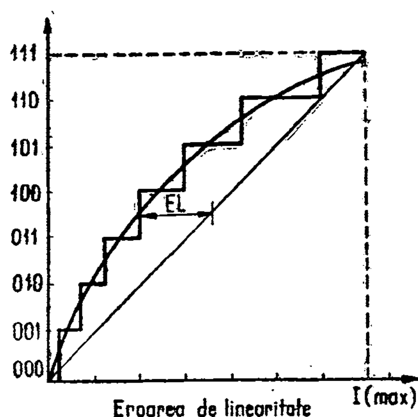


Fig. 9.18. Eroarea de linearitate.

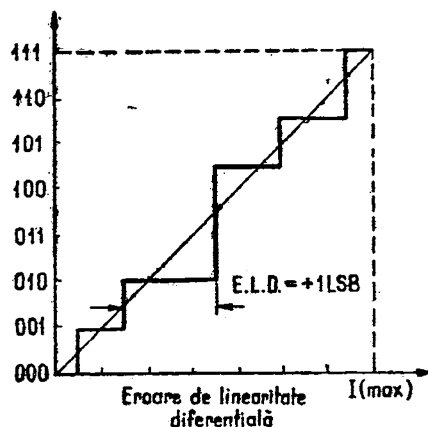


Fig. 9.19. Eroarea de linearitate diferențială.

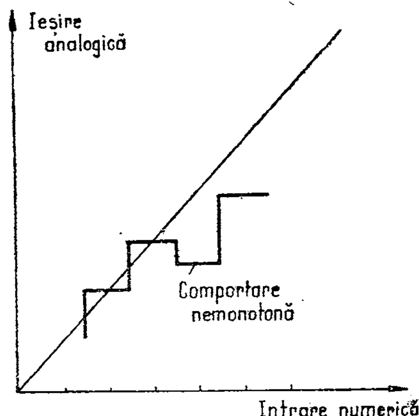


Fig. 9.20. Comportare monotonă.

de același semn cu cea a semnalului de intrare. Această caracteristică este importantă pentru funcțiile de reglare.

Timpul de conversie este intervalul necesar modulului de intrare analogică pentru executarea unei conversii.

Rata conversiei este exprimată de numărul de conversii pe secundă și măsoară viteza cu care poate lucra un modul.

Coeficientul de variație cu temperatura reprezintă capacitatea modulelor de a-și menține caracteristicile nominale în condițiile modificării temperaturii de lucru. Modificarea parametrilor componentelor active și pasive duce la modificarea curenților, tensiunilor și rezistențelor, în consecință la creșterea erorii de amplificare a deplasării caracteristicii de transfer a linearității și a monotoniei.

Stabilitatea în timp reflectă capacitatea modulelor de a-și menține caracteristicile nominale pe perioade de timp definite. Aceste modificări

se datoresc fenomenelor de îmbătrînire a componentelor. Erorile de amplificare și deplasarea caracteristicii de transfer trebuie reglate periodic, pentru compensarea îmbătrînirii componentelor.

9.3.3. Module de intrări analogice

La realizarea modulelor de intrări analogice s-au avut în vedere tipurile de semnale utilizate în mod curent în țara noastră și în general în sistemele de automatizare, nivelul de zgomot emis de traductoarele de măsurare și posibilitățile de recepționare de zgomote pe liniile de transmisie a semnalelor.

S-au putut grupa tipurile de semnale recepționate în următoarele categorii, care acoperă cea mai mare parte a semnalelor utilizate în energetică, metalurgie, chimie etc. :

— semnale unificate, specifice sistemelor de automatizare cu ieșire în buclă de curent continuu (0—10 mA, 0—20 mA, 2—10 mA, 4—20 mA),

— semnale cu amplitudini de ordinul mV, generate de traductoare de tip termocuplu și echipamente nestandard,

— semnale emise de traductoare de temperatură, de tip termorezistență.

Schema bloc din figura 9.21 înfățișează părțile componente ale modulului de intrări analogice. Principal el se compune din multiplexorul de intrare, un amplificator, convertorul analog-numeric, interfața cu calculatorul și surse.

Multiplexarea este operația de comutare a cîte unui semnal de intrare la blocul de amplificare și conversie. Se poate realiza prin două tehnologii : cu relee tip reed, care au inconvenientul vitezei reduse de comutare (1÷6 ms) și cu elemente semiconductoare, care asigură viteze ridicate de comutare (pînă la 10 ns), dar care nu pot suporta tensiuni între intrări mai mari de 10÷20 V.

În sistemele SPOT 80 s-a folosit multiplexorul cu relee din considerente de fiabilitate.

Amplificatorul realizează adaptarea semnalelor pentru intrarea în convertorul analog numeric. Constructiv este realizat în așa fel încît să

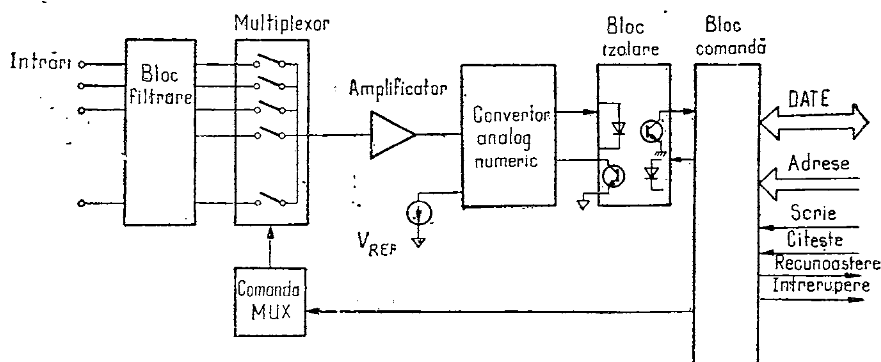


Fig. 9.21. Modul intrări analogice.

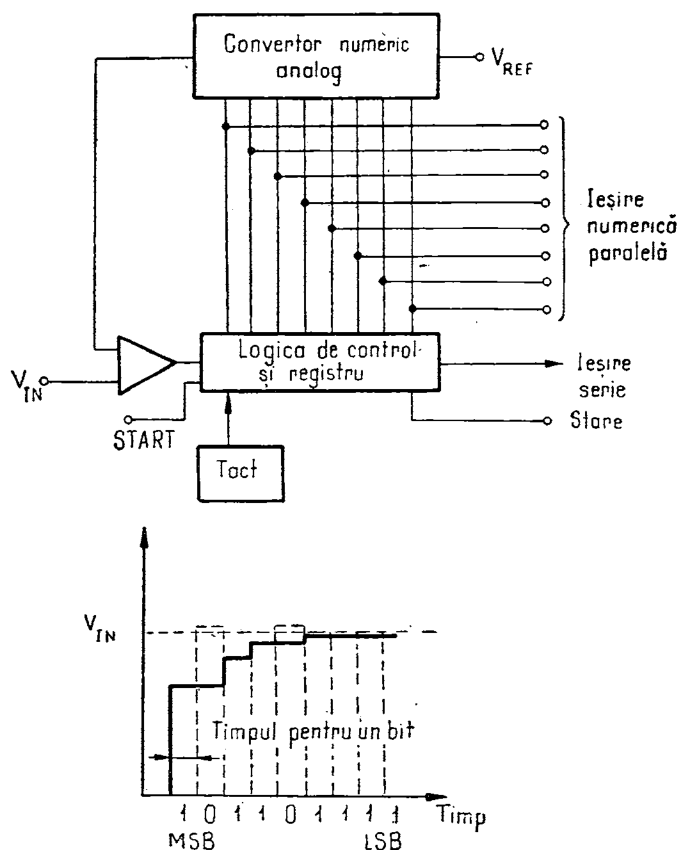


Fig. 9.22. Convertor analog-numeric cu aproximări succesive.

prezinte o impedanță ridicată la intrare și o rejecție mare de zgomot de mod comun.

Convertorul analog numeric este unul din cele mai importante componente ale modulelor de intrări analogice. El este destinat să transforme datele din forma analogică, continuu variabilă, într-un cod numeric, accesibil calculatorului numeric. Convertoarele analog-numerice sînt realizate în multe variante, în cele ce urmează se vor descrie variantele utilizate în sistemul SPOT 80.

Pentru interfațarea cu calculatorul numeric și timpul dat în care se obține valoarea numerică, se folosește convertorul cu aproximări succesive. Conversia este independentă în timp de valoarea tensiunii de intrare, fiecare conversie este unică și independentă de rezultatul conversiei anterioare, logica internă este inițiată și repornită de fiecare dată. Tehnica de conversie (figura 9.22) constă din compararea intrării necunoscute, cu o tensiune generată intern, la ieșirea unui convertor numeric-analog. Intrarea în convertorul numeric-analog este ieșirea numerică a convertorului analog-numeric. După primirea comenzii de conversie, convertorul emite la ieșire o tensiune corespunzătoare celui mai semnifi-

cativ bit (MSB), care are valoarea de $^1/_2$ din întreaga scală și se compară cu intrarea. Cînd intrarea este mai mare se memorează în registru valoarea „1", în caz contrar valoarea „0" și se trece la compararea următorului bit egal cu $^1/_4$ din scală. Procesul continuă pînă la compararea ultimului bit al conversiei, cînd în registrul de ieșire se va obține valoarea convertită.

Se observă că, pe perioada conversiei, mărimea de intrare nu trebuie să-și modifice valoarea, în caz contrar codul obținut la ieșire va avea o valoare eronată. Pentru acest tip de convertor intrările trebuie să varieze suficient de încet, la intrare zgomotele trebuie filtrate corespunzător, pentru a nu produce modificarea intrării pe parcursul conversiei. În cele mai multe situații acest convertor se folosește cu un circuit de eșantionare și memorare a mărimii analogice la intrare.

În cazul semnalelor de nivel mic, raportul semnal-zgomot capătă valori mici și sînt necesare precauții speciale pentru obținerea unei informații cît mai precise. Pentru modulele destinate măsurării acestor semnale s-a preferat utilizarea convertoarelor tip dublă rampă. Acestea necesită un timp relativ lung de conversie, pe parcursul căruia se realizează o filtrare bună a frecvențelor parazite.

Semnalul de intrare (figura 9.23) se aplică pe un integrator în același timp cu pornirea unui numărător. După un număr stabilit de impulsuri (ceea ce reprezintă un timp determinat), se aplică la intrare o tensiune de polaritate inversă, considerată etalon. Condensatorul de integrare se va descărca într-un timp proporțional cu valoarea medie a tensiunii de la intrare. În același moment cu cuplarea tensiunii etalon, se pornește din zero și un numărător, care se oprește cînd comparatorul de la ieșire va sesiza valoarea zero. Conținutul numărătorului reprezintă valoarea convertită a intrării.

Calitățile conversiei dublă rampă sînt : precizia independentă de valoarea condensatorului și frecvența de măsurare, linearitatea foarte bună, toate codurile sînt posibile, integrarea reduce zgomotul de înaltă frecvență și realizează o măsurare medie. La construirea modulelor de intrări analogice, pentru semnal mic s-a folosit sincronizarea integrării cu un număr întreg de alternanțe din rețeaua de forță, pentru rejecția zgomotelor recepționate de la instalațiile de forță.

Timpul de conversie al modulelor cu convertoare dublă rampă este determinat de frecvența fundamentală ce trebuie rejectată. În cazul rețelei de 50 Hz, pentru sincronizarea cu o singură alternanță, timpul de conversie, care conține și timpul de comutare, este de 50 ms.

În traductorul de emisie al semnalului de măsurare și pe căile lui de conectare, se produce interferența electromagnetică cu alte fenomene. Acestea se manifestă la intrarea calculatorului sub formă de zgomote suprapuse semnalului util. Zgomotele se pot împărți în două categorii : zgomot de mod diferențial, care apare între cele două borne de la intrare și zgomotul de mod comun, care apare ca tensiune față de masă, egal pe ambele intrări.

Zgomotul diferențial se reduce în mod eficient prin utilizarea unor filtre RC echilibrate, pe fiecare intrare. Rejecția semnalului de zgomot crește exponențial cu frecvența, dar introduce și o întîrziere a semnalului util, deci o limitare în viteza de variație a acestuia.

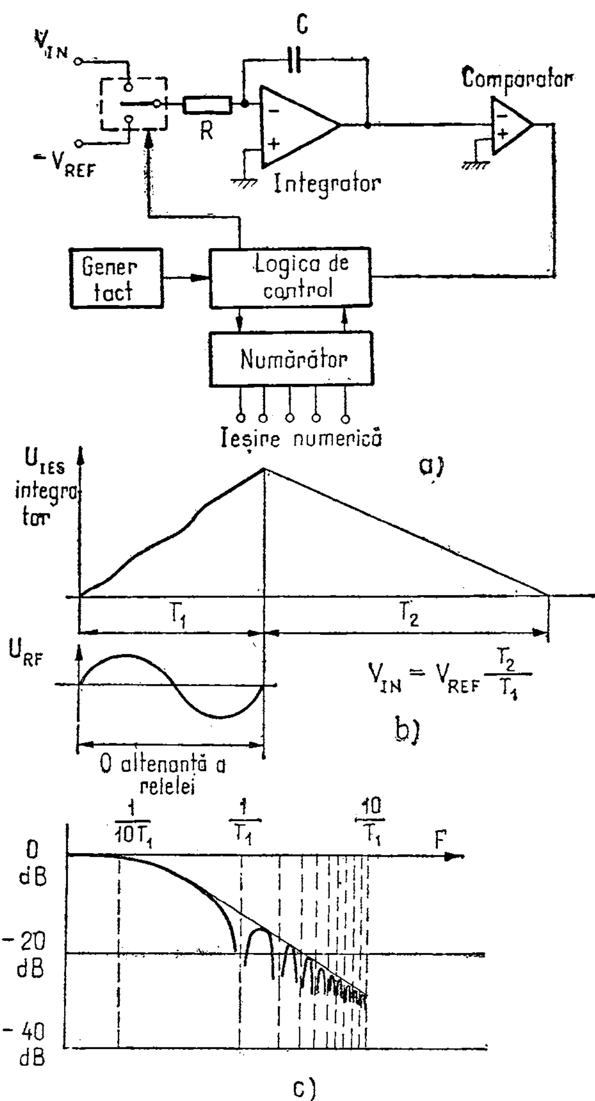


Fig. 9.23. Convertor analog-numeric dublă rampă, schema de principiu (a), diagrama de timp (b), rejecția de zgomot (c).

Zgomotul de mod comun se reduce prin echilibrarea intrărilor, utilizînd amplificatoare de intrare cu impedanță mare (intrare pe tranzistoare FET) și izolarea față de masă a etajului de intrare. O izolare perfectă nu se poate obține, dar, prin aceste metode de realizare, conectarea la masă prin impedanțe de valori foarte mari reduce în mod corespunzător curenții de scurgeri.

Numărul mare de intrări ale unui modul (64) însumează impedanțe paralele suplimentare față de masă. În sistemul SPOT 80 acest neajuns se ameliorează prin multiplexare pe blocuri.

9.3.4. Modulul de ieșiri analogice

Modulul de ieșiri analogice constituie dispozitivul prin care se pot transmite date de la calculatorul numeric către proces, pentru conducerea în circuit închis. În principiu acest modul convertește cuvîntul numeric într-o mărime analogică, de obicei curenți unificați, folosiți pentru comanda elementelor de execuție.

Schema bloc simplificată (figura 9.24) reprezintă modulul de ieșiri analogice, din sistemul SPOT 80. Modulul comandă 8 linii de ieșire, fiecare izolată electric de celelalte și de procesorul central. Fiecare linie memorează mărimea de ieșire pe un registru, la ieșirea căruia se află un convertor numeric-analogic și un amplificator operațional, pentru adaptarea semnalului. Pentru izolarea galvanică a fiecărei ieșiri, registrele de memorare sînt încărcate prin optoculuare. Ieșirea paralelă a acestui registru reprezintă intrarea pentru convertorul numeric-analogic.

Pe timpul transmisiei, ieșirea din convertorul numeric-analogic este decuplată de amplificator, pentru a nu produce perturbarea semnalului de ieșire. În acest interval, valoarea anterioară a ieșirii convertorului numeric-analogic este menținută la intrarea amplificatorului printr-un condensator.

Convertorul numeric-analogic (figura 9.25) acceptă cuvinte numerice la intrare și le transformă în tensiuni. Valoarea ieșirii unui astfel de convertor este dată de expresia :

$$Ui = Vr\ (a_n 2^n + a_{n-1} 2^{n-1} + \ldots + a_0 2^0),$$

unde Vr este tensiunea de referință, iar a_n, $a_{n-1}, \ldots$, a_0 sînt coeficienții corespunzători reprezentării numerice, care primesc valoarea 1 pentru „1“ logic și valoarea 0 pentru „0“ logic, în cuvîntul numeric. Bitul MSB va avea valoarea Vr/2 iar LSB Vr/2.

În convertorul utilizat, în sistemul SPOT 80, fiecare bit comandă un generator de curent constant, conectat la nodurile unei rețele de re-

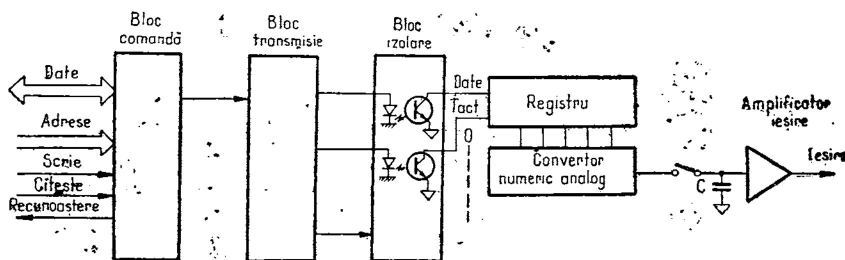


Fig. 9.24. Modul ieșiri analogice.

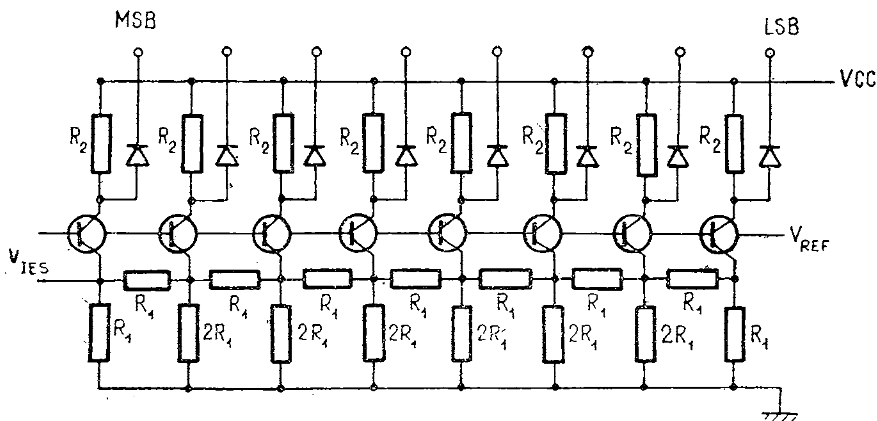


Fig. 9.25. Convertor numeric-analogic.

zistențe de tip R-2R. Tensiunile de alimentare ale rețelei sînt astfel alese încît să poată fi comandate direct cu niveluri TTL.

9.3.5. Modulul de intrări numerice

Multe traductoare emit semnale cu două stări discrete : contacte de relee sau dispozitive cu două nivele de tensiune la ieșire. Semnalele cu două stări pot proveni de la traductoare optice, mecanice sau inductive de rotație, de la traductoare de poziție, supape de siguranță etc. Aceste semnale sînt convertite în modulul de intrări numerice în valori „0" sau „1" logice, asociate cu adresa liniei pe care au sosit.

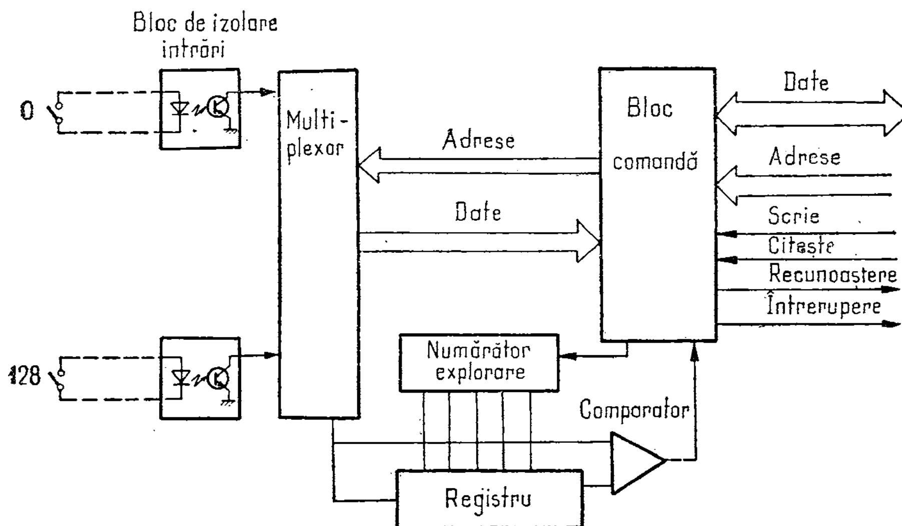


Fig. 9.26. Modul intrări numerice.

Modulul de intrări numerice are două funcţii : una de explorare continuă a stării tuturor intrărilor şi emiterea unui semnal către procesorul central la sesizarea unei modificări şi a doua, de citirea stării intrărilor la cerere.

Schema bloc din figura 9.26 reprezintă blocul circuitelor de intrare, registrul de explorare serială şi comparare şi blocul de cuplare la procesorul central.

Organizarea modulului este realizată în grupe de cîte 8 intrări. La cererea procesorului central, exprimată prin trimiterea adresei grupului de la care se cer informaţii, explorarea intrărilor se opreşte şi se reia numai după primirea semnalului de citire a datelor cerute.

Pe liniile de intrări numerice apar impulsuri parazite, care pot să creeze semnale false sau să elimine semnale adevărate, de aceea s-au luat măsuri speciale de protecţie. Fiecare intrare este izolată prin cuploare optice, realizînd practic o anulare a zgomotului de mod comun, curentul, reprezentînd starea zero logic, este de 20—50 mA şi constituie un prag suficient de mare pentru zgomotele diferenţiale.

9.3.6. Modulul de ieşiri numerice

Modulul de ieşiri numerice are sarcina de a emite semnale de comandă pentru pornirea sau oprirea motoarelor, cuplarea liniilor de transport a energiei, închiderea şi deschiderea ventilelor, comanda lămpilor de semnalizare etc.

În figura 9.27 este dată schema bloc a modulului de ieşiri numerice, din calculatorul de proces SPOT 80.

Ieşirile, în număr de 128, sînt împărţite în 16 grupe, de cîte 8 linii. Ele sînt realizate în două variante : una cu comandă pe relee, care permite comutarea unor tensiuni de 100 V.c.a. şi 0,5 A, dar cu viteză mică de comutare, de maximum 500 μs. şi a doua cu ieşire prin cuplare optică şi tranzistor, cu colectorul deschis, care poate comuta 48 V.c.c. şi 0,1 A, dar cu o viteză de comutare de 100 ns.

Comanda unei ieşiri se realizează prin emiterea de către procesor a două instrucţiuni : prima, care transmite adresa grupei şi a doua, care transmite noua stare, a celor 8 ieşiri, din grupă.

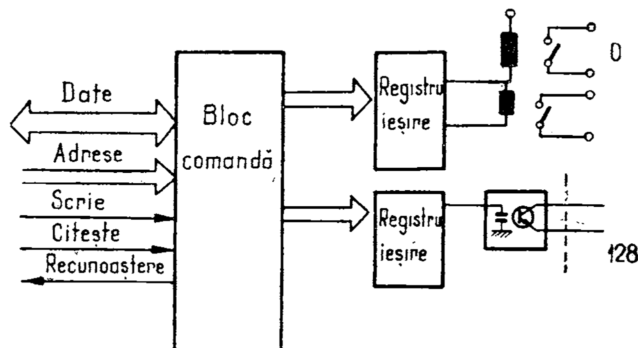


Fig. 9.27. Modul ieşiri numerice.

9.3.7. Calculatorul pentru conducerea procesului de electroliză

Reducerea electrolitică a aluminei se realizează cu ajutorul unei linii de cuve de electroliză, legate în serie. În fiecare cuvă se află un electrod de carbon (anod) scufundat într-o baie de criolit topit, cu funcția de electrolit (3NaF.AlF), în care se încarcă apoi alumina (AlO). Prin efect de electroliză se produce aluminiul, care se depune pe fundul cuvei, unde joacă rol de catod și se extrage periodic. La o astfel de linie consumul de energie atinge valori foarte mari. În regim nominal, pe fiecare cuvă se repartizează circa 3,9 V, iar curentul capătă valori, funcție de varianta tehnologică, de la 65 000 A, la peste 150 000 A, la construcțiile moderne. Tensiunea pe cuvă depinde de distanța între anod și catod. Prin folosirea unei distanțe corespunzătoare, se realizează un consum de energie optim. În cuvă se menține permanent o concentrație de alumină dictată de proces. O concentrație slabă produce ca efect parazit electroliza Al.F, care eliberează fluorina și polarizează anodul (efectul anodic). În astfel de situații, tensiunea pe cuvă crește pînă la valori de 50 V, iar energia debitată suplimentar are efecte dăunătoare asupra cuvei și electrozilor. Calculatorul de proces capătă, în aceste condiții, un efect economic important. El calculează rezistența electrică totală a fiecărei cuve (prin măsurarea tensiunii pe cuvă și a curentului pe linie) și dă comenzile necesare pentru eliminarea situațiilor de ieșire din parametrii nominali. Prin observarea modului de variație a rezistenței pe cuvă, calculatorul determină concentrația de alumină și semnalizează necesitatea încărcării.

Sistemul pentru conducerea unei linii de 256 cuve este compus dintr-un minicalculator de tip CORAL, cu sarcina de a superviza funcționarea a 16 microcalculatoare de proces și a permite accesul operatorului în stabilirea parametrilor de funcționare ai cuvelor. Minicalculatorul întocmește jurnale zilnice de funcționare, stabilește consumul de energie, evidențiază pierderile de energie datorate efectelor parazite etc.

Fiecare microcalculator supraveghează funcționarea a 16 cuve, măsurînd tensiunea pe cuva și curentul serie, calculînd rezistența pe cuvă, studiind tendința de variație a tensiunii, și efectuează operațiile de comandă referitoare la creșterea sau scăderea tensiunii pe cuvă, prin modificarea distanței anod-catod. Sesizează apropierea de efectul anodic și ia măsuri pentru eliminarea acestuia.

Constructiv, microcalculatorul (figura 9.28) este compus dintr-o unitate centrală din sistemul FELIX M18, 118, bazată pe microprocesorul 8080, cu comunicare serială la minicalculatorul supervizor, o a doua ieșire serială pentru posibilitatea conectării unei console, un ceas de timp real, sistem de priorități al întreruperilor, memorie fixă pentru programe de 16 ko, memorie de date de 2 ko, un modul pentru conversia tensiunilor, module pentru intrări numerice și module pentru comenzi.

Sistemul de întreruperi are conectate, pe nivelul de prioritate zero, semnale corespunzătoare erorii de paritate a memoriei, eroarea de apelare a unui periferic, semnalul de cădere a tensiunii de alimentare, semnalul de supraveghere a funcționării corecte a programului. Pe ni-

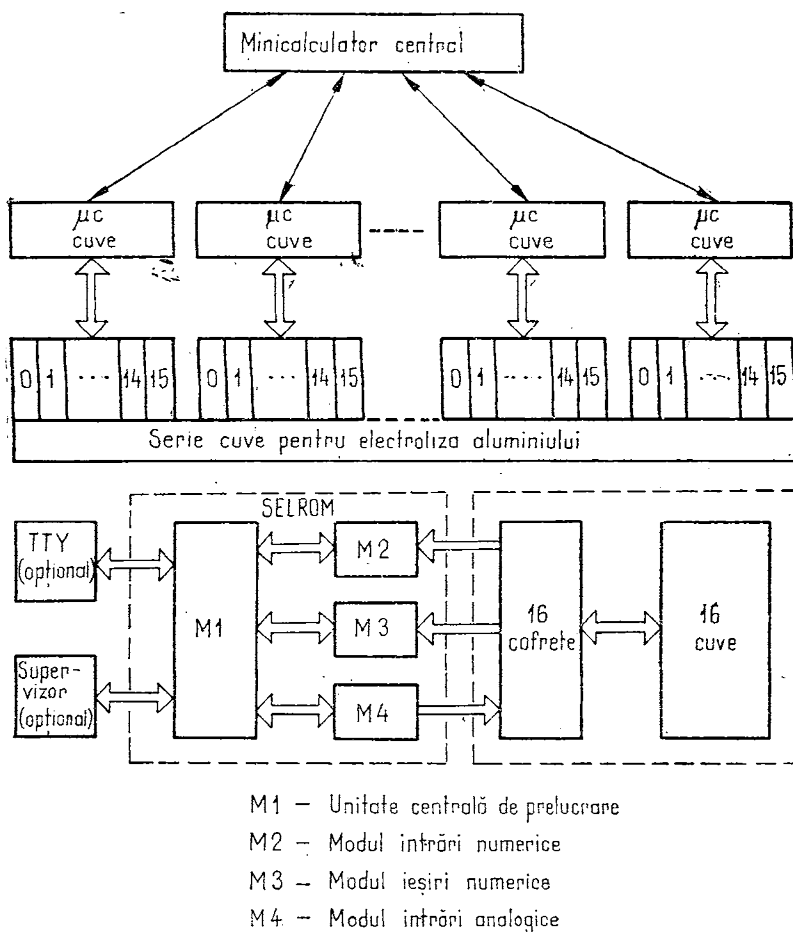


Fig. 9.28. Conducerea procesului de electroliză a aluminei.

velul 1 de prioritate este conectat ceasul de timp real. Pe nivelurile 2÷3 de prioritate se conectează intrările numerice. Pe nivelul 4 de prioritate se plasează semnalul de terminare a conversiei modulului de măsurare a tensiunilor. Pe nivelurile 5—6 sînt cuplate cele două legături seriale, iar nivelul 7 este mascat. Pentru nivelul 7 de prioritate s-au scris totuși programe, deoarece pe acest nivel sînt semnalizate zgomotele în sistemul de întreruperi.

Pentru conectarea cu procesul, fiecare bloc de conducere este dotat cu următoarele module de intrare/ieșire :

Modulul de intrări analogice care convertește tensiunile, de la 16 cuve, în cuvinte de 10 biți. Multiplexarea acestor tensiuni se realizează în prezența unor curenți de scurgeri la masă, distribuiți de la fiecare cuvă. Curenții de scurgere fac să se obțină o tensiune de mod

comun la fiecare intrare, variabilă de la un modul la altul, în domeniul +/—600 V c.c. În condiții limită, cînd una din cuvele aflată la unul din capete are scurgeri de aluminiu la masă, modulele de la celălalt capăt al liniei trebuie să suporte tensiuni de mod comun de 1 200—1 500 V c.c. Din acest motiv multiplexorul s-a realizat cu deosebită grijă, din relee cu mercur, care asigură rigiditatea dielectrică cerută.

Măsurarea tensiunilor se realizează pe două scale de 100 V și 10 V, viteza de conversie fiind de 40—50 ms. Comutarea celor două scale se realizează prin program.

Modulul de intrări numerice supraveghează 64 contacte, provenite de la tablourile de cuvă. El este utilizat în ambele moduri de funcționare : explorarea intrărilor, emiterea unei întreruperi la întîlnirea unei modificări și citirea intrărilor la cerere.

Modulul de ieșiri analogice comandă 80 de linii, grupate în 4 comenzi pe fiecare cuvă și 16 semnalizări, pe panoul frontal al calculatorului.

9.3.8. Calculator pentru conducerea procesului de galvanizare

Procesul de galvanizare (figura 9.29) constă din depunerea unei anumite cantități de metal în mod uniform pe suprafața pieselor. Cantitatea de metal depus pe unitatea de suprafață depinde de : densitatea de curent în băile de acoperire, de eficiența catozilor și a electrolitului, de temperatura în baie și calitatea suprafeței înainte de acoperire.

Pentru aducerea suprafețelor la starea corespunzătoare acoperirii, piesele sînt supuse la repetate spălări și degresări. Fluxul tehnologic este format dintr-o succesiune de băi cu funcții determinate.

Sarcina calculatorului de proces constă în poziționarea transportului de piese în fiecare baie, conform procesului tehnologic, menținerea în fiecare baie, un timp corespunzător operației efectuate, supravegherea nivelului de lichid, reglarea temperaturii, reglarea densității de curent în băile de depunere, măsurarea și reglarea conținutului de săruri în fiecare baie, acționarea jeturilor de apă de spălare numai în perioada existenței pieselor etc.

Sistemul SPOT 80, folosit în acest scop, este compus din : procesorul central, cu memorie de program de 4 ko, memorie RAM 8 ko, ceas de timp real, sistem de întreruperi, transmisie serială pentru consolă, modulul de intrări analogice (necesar pentru măsurarea temperaturilor și densității de curent), modul de intrări numerice (pentru citirea poziției manipulatorului, a comenzilor operatorului, a valorilor prestabilite, a nivelurilor în băi), modul de ieșiri numerice (pentru comanda motoarelor de deplasare, a electroventilelor, a electropompelor) și modulul de ieșiri analogice, pentru comanda redresoarelor.

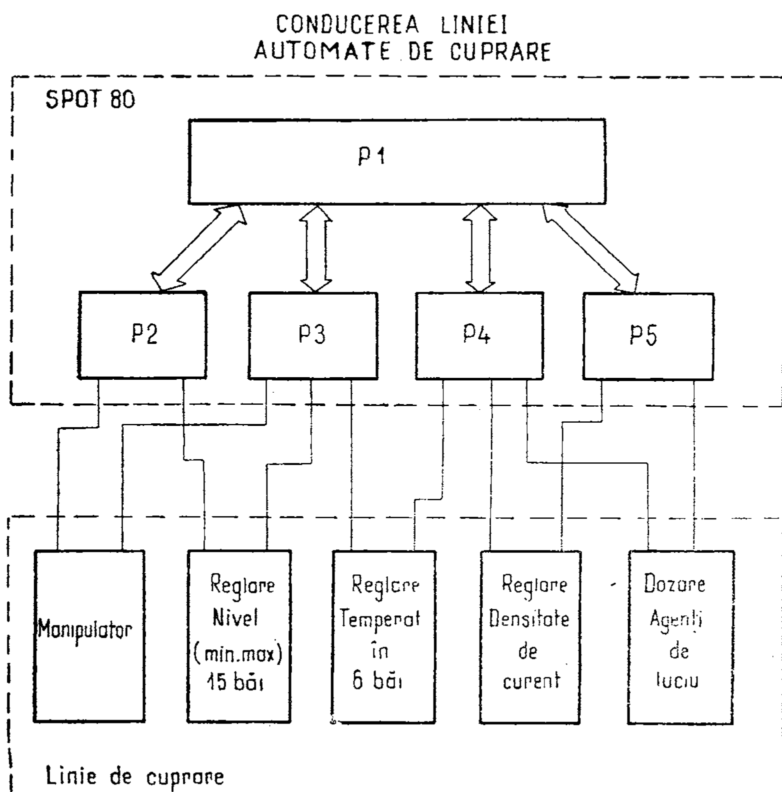


Fig. 9.29. Conducerea liniei automate de cuprare.

9.4. Microcalculatoarele M 18, M 118 în centrale termoelectrice

9.4.1. Elemente caracteristice conducerii proceselor termoenergetice

Procesele termoenergetice sînt procese tehnologice complexe, caracterizate de un număr mare de variabile cu interconectare multiplă, o dinamică cu întîrzieri mari de timp, un puternic caracter neliniar și în unele cazuri cu o modificare în timp a parametrilor dinamici.

Tehnologia analogică-discretă, folosită de echipamentele de automatizare convenționale, se dovedește a fi depășită pentru marile unități energetice, deoarece conduce la un volum mare de echipamente, la prețuri ridicate, indicatori de fiabilitate reduși și posibilități tehnice și ergonomice insuficiente pentru conducerea procesului.

În prezent există o disproporție între creșterea continuă a complexității instalației tehnologice cazan-turbină-generator și ansamblul sistemelor de automatizare realizate în tehnologia discretă, voluminoasă, scumpă și cu grad redus de fiabilitate.

Progresele deosebite realizate pe plan teoretic în teoria sistemelor și în știința conducerii în general s-au manifestat timid în practica industrială, unde se folosesc în continuare un volum mare de date, dar relativ sărac în informații și legi de reglare simple aplicate la un proces complex, neliniar și cu parametrii distribuiți.

Progresele tehnologice realizate în domeniul tehnicii de calcul se manifestă în prezent și în domeniul aplicațiilor industriale, unde se impune cu precădere o urgentă „împachetare" a hardware-ului sistemelor de conducere, trecerea la algoritmi complecși, bazați pe modele multivariabile și la folosirea unor variabile cu conținut îmbogățit de informații.

Procesul clasic de transformare a energiei chimice a combustibililor fosili în energie electrică, presupune existența unui flux energetic, rezultat ca urmare a procesului de ardere cu degajare de căldură și producere a gazelor de ardere cu temperatură ridicată, a unui flux masic al agentului purtător apă-abur cu schimbare de fază și un transformator de energie cu conversie din energie potențială în energie cinetică și ulterior electrică.

Fluxul energetic primar al gazelor fierbinți prin transfer de căldură, încarcă cu energie potențială fluxul masic al agentului purtător.

Acest proces are loc în cazanul de abur, iar procesul de transformare al energiei are loc în grupul turbogenerator, unde aburul de înaltă presiune și temperatură, prin procesul de destindere pune în mișcare turbina și, implicit, generatorul de energie electrică.

Revenirea la faza inițială a agentului purtător se face în condensatorul de abur, de unde condensatul obținut parcurge mai multe trepte de ridicare a presiunii și revine la intrarea în cazan, închizîndu-se astfel ciclul termic (fig. 9.30).

Controlul unui astfel de proces complex nu se poate face decît adăugînd la cele două fluxuri menționate și un flux informațional pe baza căruia să se poată lua deciziile de conducere corespunzătoare.

Fluxul informațional are o structură diversă și cuprinde ca mărimi măsurate : temperaturi, presiuni, debite, niveluri, mărimi mecanice, analitice, electrice precum și mărimi indirecte rezultate din calculul mai multor mărimi măsurabile.

Funcția scop pe care trebuie să o îndeplinească sistemul de conducere asociat unei instalații termoenergetice este asigurarea *funcționării sigure și economice* a grupurilor cazan-turbină-generator, adică, producerea continuă și fără defecțiuni, pe o perioadă de timp determinată, a energiei electrice cerute, folosind o cantitate minimă de combustibil primar și asigurînd condițiile de calitate impuse.

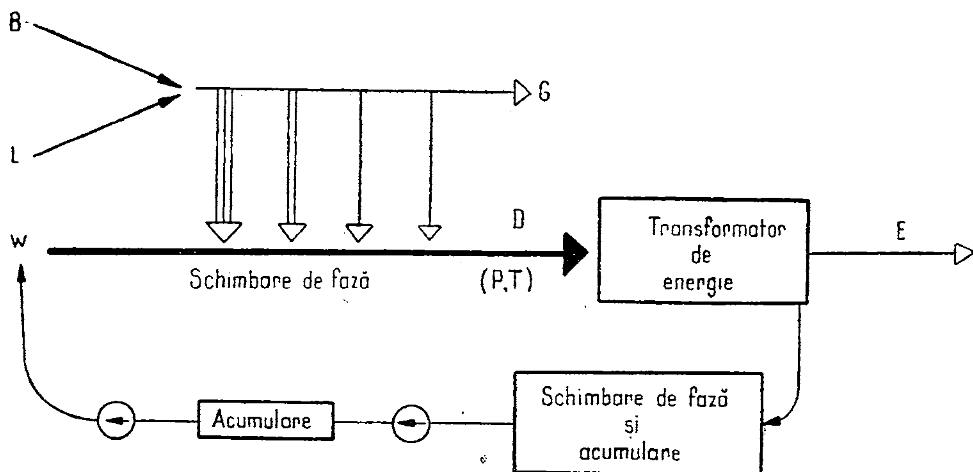


Fig. 9.30. Structuralizarea procesului de obținere a energiei electrice prin arderea combustibililor fosili.

Funcționarea economică a unui grup termoenergetic cazan-turbină-generator este producerea energiei electrice cerute, cu ajutorul unei cantității minime de combustibil primar, ceea ce este echivalent cu minimizarea consumului specific de combustibil necesar producerii unui kWh de energie electrică sau cu maximizarea randamentului global al ciclului termic.

Urmărirea economicității procesului de obținere a energiei electrice din energie primară a combustibililor fosili se poate face prin *metoda directă* a bilanțului energetic sau prin *metoda indirectă,* care implică identificarea individuală a tuturor pierderilor ce intervin în ciclul termic.

Funcționarea sigură a unui grup termoenergetic cazan-turbină-generator este definită prin probabilitatea de funcționare fără defecțiuni, într-un interval de timp considerat și în condiții date, asigurînd îndeplinirea scopului pentru care au fost proiectate.

Siguranța în funcționare a instalațiilor termoenergetice ridică probleme deosebit de complexe, ca urmare a unui complet de factori mecanici, termici, electrici și chimici.

Continuitatea funcționării și producerii energiei electrice trebuie asigurată în condițiile în care acționează asupra echipamentului termomecanic factori ca : temperaturi, presiuni, forțe centrifuge, solicitări electrice, eroziuni mecanice sau coroziuni chimice.

De asemenea, trebuie asigurate regimuri stricte de răcire, lubrefiere, etanșare și puritate a mediilor și fluidelor folosite.

Toate aceste condiții specifice proceselor termoenergetice sînt, de fapt, argumente pentru a folosi o nouă tehnologie de conducere care să aducă schimbări de fond, atît în privința echipamentelor, cît și în tehnica de conducere.

9.4.2. Structuri descentralizate de microcalculatoare — o soluție modernă pentru conducerea blocurilor de mare putere

Conducerea unui sistem energetic interconectat este organizată ca o structură ierarhică pe nivele de conducere, în care sistemul de conducere al grupurilor energetice reprezintă nivelul de bază.

În cadrul acestui nivel de conducere, linia modernă de utilizare a sistemului de calcul este descentralizarea echipamentului în module distincte, cărora li se atribuie funcțiuni de sine stătătoare.

Se realizează, astfel, sisteme de conducere la care căderile parțiale de echipament sau chiar de programe, conduc la nerealizarea numai a unor funcțiuni și nu la scoaterea completă din funcțiune a întregului ansamblu.

Pentru funcțiunile foarte importante, cum sînt cele de comandă sau protecție, se pot folosi metodele de rezervare activă prin dublarea elementelor și asigurarea funcționării în paralel cu preluarea din mers a funcțiunii respective în cazul unei căderi.

Organizarea structurilor descentralizate de minicalculatoare este determinată de interinfluențarea a cinci criterii fundamentale, și anume : criteriul funcțional, geografic, tehnologic, de fiabilitate și economic.

Criteriul funcțional este criteriul fundamental în alegerea structurii sistemului de conducere și se referă la proprietatea de decompozabilitate a funcției scop prezentată în capitolul anterior, în funcții elementare cvasiautonome, ce pot fi atribuite unor subsisteme ale echipamentelor de conducere între care comunicarea să fie minimă.

În acest fel, rezultă o structură descentralizată funcțional, care lucrează totuși ca o structură multiprocesor, datorită acestui minim de legături care se păstrează între subsistemele echipamentului de conducere.

Funcția scop a grupului energetic o vom numi în continuare *funcția globală de conducere*, **F**, și care în formulare restrînsă este producerea continuă și fără defecțiuni, pe o perioadă de timp determinată, a energiei electrice cerute, folosind o cantitate minimă de combustibil primar și asigurînd condițiile de calitate impuse.

Funcția globală de conducere, în general, se poate descompune într-o componentă informațională, F_I și o componentă de comandă, F_C.

Criteriul geografic impune restricții legate de amplasarea în teren a diferitelor subsisteme tehnologice, de posibilitățile de transmitere și comunicare a informațiilor și de posibilitățile de transmitere a comenzilor.

Criteriul tehnologic introduce restricții, ca urmare a capacității tehnice limitate a echipamentelor de calcul și a necesității de a alege un optim între volumul, viteza și costul schimbului de informații dintre diferitele subsisteme distribuite.

Caracterul restrictiv al criteriului tehnologic este într-o dinamică permanentă, progresele tehnologice mărind continuu posibilitățile tehnice ale microprocesoarelor și ale componentelor electronice integrate pe scară largă.

Criteriul de fiabilitate impune, în funcție de importanța comparativă a funcțiilor elementare pe care trebuie să le îndeplinească diferitele sub-

sisteme de conducere, gradul de rezervare, astfel încît structura rezultată să asigure o fiabilitate impusă.

Criteriul economic este cel care decide în final soluția aleasă, căutîndu-se un optim între performanțe tehnice, fiabilitate și costul total al sistemului de conducere.

Revenind la criteriul funcțional, pentru a prezenta mai multe amănunte, trebuie menționat că în cazul grupurilor termoenergetice, decompozabilitatea funcției de conducere poate continua, în funcție de cele trei regimuri de funcționare în care acestea se pot afla, adică, regimul de pornire, de exploatare în sarcină sau de avarie.

Vom da, în cele ce urmează, cîteva exemple pentru regimul de exploatare în sarcină al grupului energetic.

Funcția informațională, F_{IEX}, este o funcție decompozabilă pe sisteme și subsisteme tehnologice, în funcție de scopul pe care îl îndeplinește și de regimul de funcționare în care se află instalația tehnologică.

Funcția informațională parametrică, F_{IPAR}, este funcția cu ajutorul căreia se caracterizează *starea instalației* la un moment dat, deoarece asigură în mod operativ informații privind valorile absolute ale variabilelor tehnologice, ieșirile din limitele prescrise, depășirea vitezelor de variație ale acestora etc., afișarea pe display-uri sau miniimprimante.

Funcția economică operativă, F_{EO}, pe o durată de 1, 8 și 24 ore, folosește metoda identificării pierderilor pe cauze și pune la dispoziția operatorului o serie de informații cu caracter economic pe baza cărora se pot lua decizii pentru optimiziarea funcționării grupului energetic.

Funcția de fiabilitate operativă, F_{FO}, se compune din mai multe funcții elementare, calculate pe baza unor valori instantanee ale variabilelor măsurate cu perioade mici de timp (2÷4 s), ce pun în evidență siguranța în funcționare a unor subansamble ale instalației tehnologice.

Aceste trei funcții întregesc *funcția informațională operativă*, F_{IO}, care permite luarea deciziilor pentru conducerea operativă a blocului pe termen scurt.

$$F_{IO}=F_{IPAR}+F_{EO}+F_{FO}$$

Pentru o durată medie de funcționare a instalației tehnologice se pot defini următoarele funcții elementare informaționale :

Funcția economică pe durată medie, F_{EM}, este bilanțul economic al funcționării pe luna anterioară și reprezintă un post-calcul cu caracter de evidență statistică și de planificare.

Funcția de fiabilitate preventivă, F_{FP}, este funcția de estimare a evoluției în timp a stării agregatelor și a materialelor componente, în vederea planificării științifice a activității de reparații și întreținere a echipamentelor tehnologice.

Funcția de comunicare, F_{COM}, este cea care realizează transmiterea informațiilor între subsistemele de conducere de la același nivel și cu sistemele ierarhice superioare.

Ultimile trei funcții menționate formează *funcția informațională statistică și de comunicare*, F_{ISC}, care permite luarea deciziilor pentru conducerea blocului energetic pe termen mediu.

$$F_{ISC}=F_{EM}+F_{FP}+F_{COM}$$

Deci, funcția informațională pentru regimul de exploatare în sarcină se poate scrie *simbolic* sub forma :

$$F_{IEX}=F_{IO}+F_{ISC} \qquad \text{sau}$$

$$F_{IEX}=F_{IPAR}+F_{EO}+F_{FO}+F_{EM}+F_{FP}+F_{COM}$$

Funcția de comandă în regim de exploatare în sarcină, F_{CEX}, este funcția care realizează automat sau prin intermediul operatorului, în situațiile în care echipamentul de comandă automată este defect, cele patru condiții fundamentale pentru conducerea sigură și economică a blocului.

A. *Condiția cantitativă a echilibrului energetic.*

CONDIȚIA 1 : „Egalitate permanentă între energia cerută de sistem, respectiv, livrată de generator, și energia dezvoltată în focar prin arderea combustibilului, micșorată de pierderile de energie care însoțesc acest proces".

$$860\,N=Q_{Ba}\cdot\eta_{KT}$$

B. *Condiția calitativă a procesului de transformare energetic.*

B.1. *Condiția de calitate a procesului de ardere.*

CONDIȚIA 2 : „Arderea combustibilului în condiții optime prin respectarea în permanență a raportului corespunzător dintre cantitatea de combustibil (B) introdus în focar și debitul de aer necesar arderii (L)".

$$Q_F=F\,(B,\,L)\ \left|\ \begin{array}{l} Q_F \text{ dat} \\ \alpha_{opt}\Rightarrow B_{min}\end{array}\right.$$

Considerînd „F" o funcție parametrică ce depinde, prin proiectare de geometria focarului și de cinetica arderii, rezultă că trebuie obținut raportul optim combustibil-aer, astfel încît debitul de combustibil să fie minim la o sarcină termică a focarului dată.

B.2. *Condiția de calitate a agentului purtător.*

CONDIȚIA 3 : „Egalitate permanentă între fluxul energetic al gazelor de ardere (Q_F) și fluxul masic al agentului purtător (W), asigurînd astfel constanța parametrilor de stare ai aburului de-a lungul traseului pînă la intrarea în turbină".

$$\left(\frac{Q_F}{W}=ct,\ P=ct\right)\Rightarrow T=ct$$

Considerînd procesul din cazanul de abur ca avînd loc la presiune constantă, apare imediat implicația că raportul dintre fluxul energetic și fluxul masic al agentului purtător determină regimul de temperaturi pe circuitul apă-abur.

C. *Condiția de siguranță a procesului de transformare energetic.*

CONDIȚIA 4 : „Limitarea valorilor ce caracterizează starea instalației tehnologice și a vitezei de variație a acestora, în regim de pornire și de exploatare în sarcină, în scopul prevenirii oricăror regimuri de avarie".

$$\left(P_i;\ \frac{dP_i}{dt}\right)<\left(\overline{P_i};\ \overline{\frac{dP_i}{dt}}\right)$$

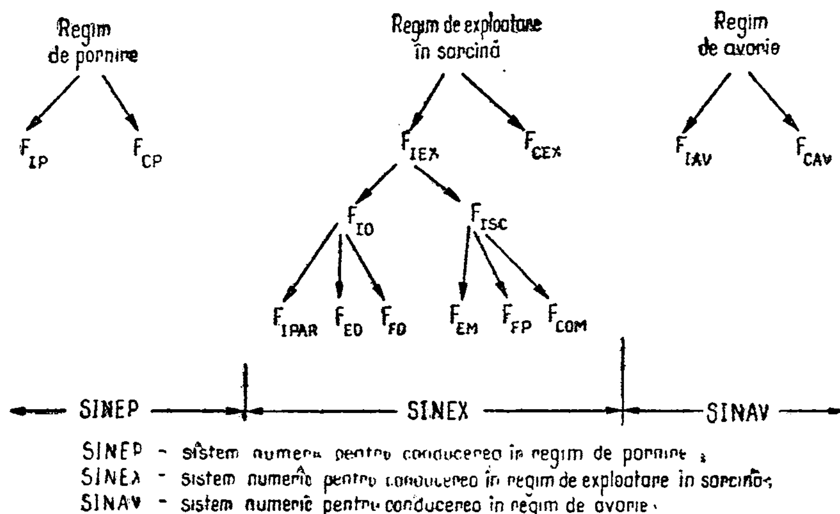


Fig. 9.31. Funcțiile elementare pentru regimul de exploatare în sarcină.

Pentru a se asigura funcționarea continuă, fără defecțiuni, este necesar să nu se depășească nici valoarea absolută și nici viteza de variație a anumitor valori prescrise pentru parametrii de control ai stării instalației.

Schematic, descentralizarea funcției de conducere pentru un grup termoenergetic, se poate reprezenta ca în fig. 9.31, în care s-au scos în evidență funcțiile elementare pentru regimul de exploatare în sarcină și s-au folosit notațiile :

F_{IP}, F_{IEX}, F_{IAV} — funcția informațională în regim de pornire, exploatare în sarcină și de avarie ;

F_{CP}, F_{CEX}, F_{CAV} — funcția de comandă în regim de pornire, exploatare în sarcină și de avarie.

Corespunzător cu cele trei regimuri în care se poate afla blocul energetic se pot folosi trei sisteme de conducere descentralizate, dacă celelalte criterii de organizare, în afara celui funcțional, nu impun o altă organizare a structurii.

9.4.3. Structuri specializate de echipamente bazate pe microcalculatoare

Ca urmare a colaborării dintre ICEMENERG și ICE au fost executate prototipurile unei serii unitare de echipamente de calcul de proces destinate supravegherii și conducerii grupurilor termoenergetice, folosind subansamble pe care ICE le are în fabricație de serie.

Aceste echipamente sînt concepute modular și organizate pe principiul descentralizării funcționale.

Subansamblele folosite aparțin microcalculatoarelor M-118 și sistemului de interfețe SPOT-80.

Asamblarea echipamentelor se face într-un pupitru cu caracter operativ, prevăzut cu display-uri și consolă de operator executată în structură

mozaicată pentru a aranja, în funcție de aplicație, desenul mnemotehnic al instalației tehnologice, tastele de comandă și lămpile de semnalizare.

Se va prezenta în continuare echipamentul SINEX-82 conceput pentru a fi folosit în regimul de exploatare în sarcină la grupurile de 330 MW, funcționînd pe cărbune.

9.4.3.1. *Structura generală a sistemului numeric de conducere a blocurilor termoenergetice în regim de exploatare în sarcină — SINEX-82*

SINEX-82, așa cum s-a prezentat în subcapitolul anterior, este un sistem descentralizat funcțional care realizează următoarele funcții elementare :

$$SINEX = \underbrace{F_{IPAR} + F_{EO} + F_{FO}}_{F_{IO}} + \underbrace{F_{EM} + F_{FP} + F_{COM}}_{F_{ISC}} + F_{CEX}$$

Dacă se tratează separat funcția informațională parametrică pentru informații numerice și pentru informații analogice, funcțiile din relația de mai sus se pot atribui la patru subsisteme de conducere, care au fost denumite după funcția cea mai importantă pe care o realizează, și anume :

DICREV — dispozitiv numeric de urmărire cronologică a evenimentelor ; realizează funcția de prelucrare a informațiilor numerice ;

$$F_{IO}^{N} - F_{IPAR}^{N}$$

DIMEC — dispozitiv numeric de urmărire operativă a mersului economic ; realizează funcția informațională parametrică pentru informații analogice, funcția economică operativă și funcția de fiabilitate operativă ;

$$F_{IO}^{A} = F_{IPAR}^{A} + F_{EO} + F_{FO}$$

DIREN — dispozitiv de reglare numerică a parametrilor tehnologici — F_{CEX}; realizează reglarea numerică a mai multor variabile tehnologice ;

DISCO — dispozitiv de supervizare, prelucrări statistice și comunicare au alte subsisteme și nivele de conducere — F_{ISC};
Se poate scrie simbolic :

$$SINEX = DICREV\ (F_{IO}^{N}) + DIMEC\ (F_{IO}^{A}) + DIREN\ (F_{CEX}) + DISCO\ (F_{ISC})$$

9.4.3.2. *Structura și funcțiile subsistemului DICREV*

Subsistemul DICREV cuprinde un microcalculator M-118, cu o componentă redusă, de nouă plachete, (FPUB, UCB, DRAM, REPROM, IOU, SAI, VDI, VDM, EXT, BUS), amplasate într-un sertar de M-18-B, două interfețe numerice de tip P2 a 128 contacte de intrare fiecare și o interfață de 32 ieșiri pe tranzistoare de tip P3 pentru comanda lămpilor de semnalizare ale consolei operatorului.

Cele trei interfețe de proces sînt cuplate la microcalculator cu ajutorul unei plachete de extensie a BUS-ului (EXT.BUS).

Structura prezentată a subsistemului DICREV nu este exhaustivă, ci poate fi adaptată în funcţie de dimensiunile aplicaţiei, ştiind că prin intermediul unei plachete EXT.BUS se pot cupla maximum patru module intrare-ieşire de tip SPOT, fără a fi nevoie de modulul P1, iar în caz de necesitate se pot folosi două plachete EXT.BUS.

Subsistemul DICREV nu conţine nici un modul P1 ceea ce simplifică mult activitatea de realizare a programelor şi asigură o fiabilitate îmbunătăţită echipamentului.

Consola operatorului conţine taste de comandă care permit afişarea la cerere, pe display, a configuraţiilor de evenimente numerice.

Orice schimbare de stare a unor contacte comandă automat, prin P3, led-ul de semnalizare asociat tastei cu care se apelează pagina respectivă, emiţîndu-se totodată şi un semnal sonor de avertizare.

O miniimprimantă asigură transcrierea, la cerere, pe hîrtie a informaţiilor prelucrate şi înregistrarea automată „prin întreruperi" în cazul apariţiei unor evenimente importante pentru buna funcţionare a procesului tehnologic.

Tastatura monitoarelor de display a fost înlocuită de o tastatură specializată, ce permite aranjarea pe consolă operatorului a tastelor de comandă în funcţie de necesităţi.

În cazul unor depanări sau modificări de programe, tastatura originală se poate recupla înlocuind în perioada respectivă tastatura specializată a consolei operatorului.

Dezvoltarea sistemului de programe aplicative se poate face pe un alt sistem sau chiar pe subsistemul DICREV, care are posibilitatea de introducere a plachetelor unităţii duale de floppy-disc (FBI, FPM, FDI) în sertarul de M18-B şi spaţiul corespunzător amplasării acestuia în cabinetul echipamentului, după care programul rezultat se înscrie în EPROM-uri.

Subsistemul DICREV destinat urmăririi cronologice a evenimentelor într-o centrală termoelectrică, ca şi celelalte subsisteme ale echipamentului SINEX-82, este prevăzut cu un monitor de timp real de tip RTX-SINEX.

Monitorul asigură gestionarea tuturor taskurilor în oricare din configuraţiile arătate în figura 9.31.

Sistemul de program aplicativ asigură îndeplinirea unor funcţii tipizate, dar poate fi completat sau reconfigurat în conformitate cu particularităţile aplicaţiilor.

DICREV, în varianta standard, realizează următoarele funcţii de proces :

a) funcţia de înregistrare automată, cronologică, a funcţionării protecţiilor tehnologice a grupului termoenergetic în cazul declanşării unei avarii.

Se realizează prin înregistrarea automată pe o miniimprimantă a tuturor contactelor care se modifică în cursul unei avarii, în ordinea apariţiei lor cronologice.

Buffer-ul de acumulare poate fi de 512 contacte începînd cu primul eveniment (la cerere 1024), fără ca miniimprimanta să fie în funcţiune, urmînd ca afişarea să se facă automat în momentul conectării acesteia.

În mod obişnuit însă miniimprimanta se află conectată şi asigură afişarea evenimentelor asincron cu producerea acestora, mărindu-se în acest fel capacitatea buffer-ului de acumulare ;

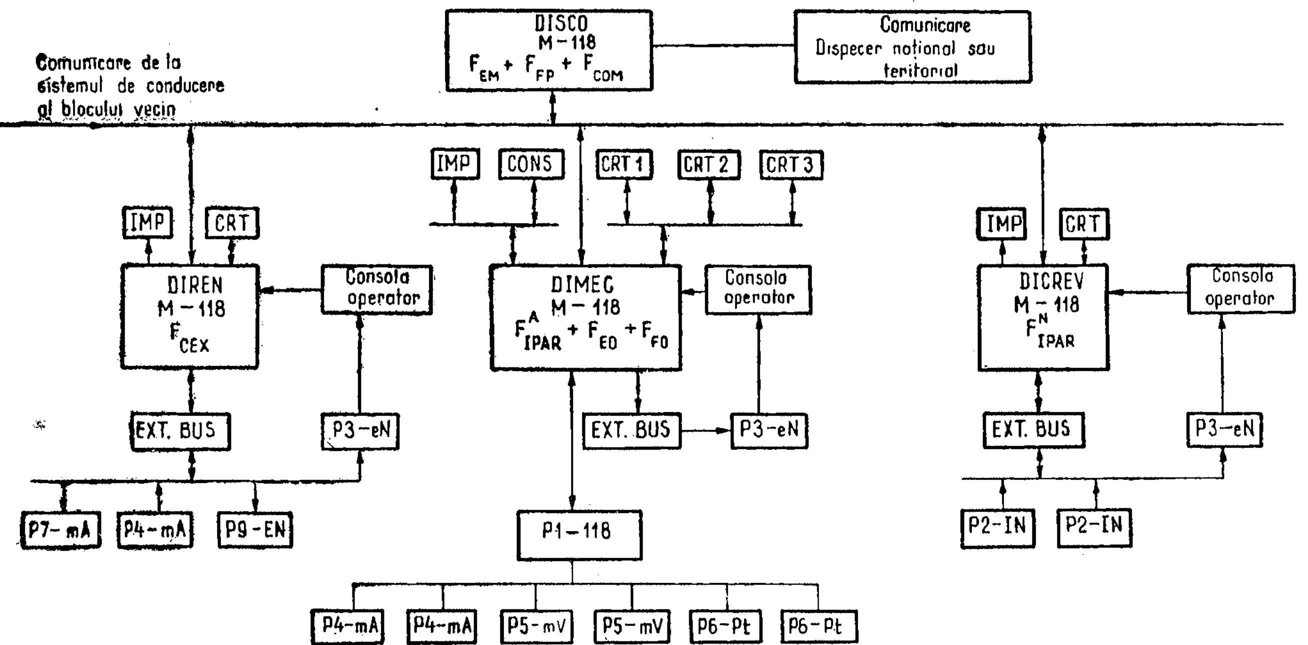


Fig. 9.32. Structura sistemului numeric de conducere în regim de exploatare în sarcina SINEX-82, pentru un grup energetic de 330 MW, funcționînd pe cărbune.

b) funcția de afișare la cerere a stării permisiilor de pornire ale agregatelor principale.

Se realizează cu ajutorul tastelor de pe consola operatorului care comandă afișarea pe display a stării permisiilor. Există taste dedicate principalelor grupe de agregate tehnologice, exemplu : PAR, VA, VG, EPA, TPA, cazan etc. ;

c) funcția de afișare la cerere a stării protecțiilor tehnologice.

Un alt grup de taste de comandă de pe consola operatorului permit afișarea la cerere pe display a stărilor la un moment dat a protecțiilor tehnologice grupate în pagini de display, corespunzător următoarelor subsisteme tehnologice : alimentare servicii interne, PAR, VA, VG, cazan, mori de cărbune, electropompe de alimentare, turbopompă de alimentare, pompe de condensat ;

d) funcția de evidență a duratei de funcționare a subansamblelor tehnologice.

Se realizează prin contorizarea la 1, 8 și 24 ore a duratei de funcționare și afișarea acestora pe display prin intermediul clapelor funcționale.

Aceste informații se folosesc în cadrul sistemului SINEX pentru calculul energiei livrate sistemului, de agregatele în funcțiune și a energiei nelivrate datorită indisponibilității unor echipamente sau subansamble tehnologice.

9.4.3.3. *Structura și funcțiile subsistemului DIMEC*

Subsistemul DIMEC are în structură un microcalculator M118 introdus într-un sertar de M18-B, șase module SPOT-80 (fig. 9.32) cuplate prin intermediul unui modul P1-118 care are rolul modulului P1 din sistemul de interfețe SPOT-80, dar este realizat pe baza unității centrale de M118.

Legătura între P1-118 și microcalculatorul M118 se face prin legătură serială.

Microcalculatorul M118 este pregătit în structură specială pentru a permite ca interfețe standard, o consolă Centronics și o miniimprimantă sau două miniimprimante, unul pînă la trei display-uri și o consolă a operatorului prevăzut cu taste de comandă, distribuite pe schema mnemotehnică a procesului și lămpi de semnalizare a limitelor fixate pentru variabilele tehnologice.

Se pot folosi pînă la 64 taste de comandă cu funcțiuni independente și aranjate după dorință, datorită structurii mozaicate a consolei operatorului, pe lîngă clapele funcționale cu care sînt prevăzute de fabrică toate display-urile.

Tastatura distribuită de proces poate fi înlocuită prin schimbarea unei cuple, cu tastatura oricărui display, atunci cînd se dorește să se facă modificări în programe.

Subsistemul poate asigura pînă la patru legături seriale sincrone sau asincrone cu echipamentele vecine și cu sistemul ierarhic superior, în afara legăturilor pe care le permite placheta SIN.

Cele două unități de calcul au în componență următoarele plachete :

M118 — FPUB, UCB, DRAM, REPROM, SAI, IOU, SIN, 3 VDI, 3 VDM.

P1-118 — FPUB, UCB, DRAM, REPROM, SAI (SIN).

DIMEC, după cum s-a arătat în paragraful 9.4.3.1, realizează funcţia informaţională operativă pentru informaţii analogice, compusă din : funcţia informaţională parametrică pentru informaţii analogice, F_{IPAR}^{A} funcţia economică operativă, F_{EO}, şi funcţia de fiabilitate operativă, F_{FO}.

Din punctul de vedere al sistemului de calcul îndeplinirea funcţiunilor menţionate înseamnă :

a) Culegerea şi prelucrarea primară a datelor; se realizează cu ajutorul modulelor din sistemul de interfeţe SPOT-80 şi a microsistemului P1-118 ;
b) Afişarea pe cele trei display-uri a unor scheme tehnologice pe care se actualizează în permanenţă, cu o periodicitate de 3—5 s variabilele caracteristice ale procesului ;
c) Înregistrarea automată pe miniimprimantă a timpului şi a valorilor tehnologice la ieşirea şi revenirea în limite ;
d) Calculul valorilor medii ale mărimilor tehnologice la 1, 8 şi 24 ore ;
e) Calculul şi afişarea indicilor de mers economic la 1, 8 şi 24 ore ;
f) Înregistrarea automată pe miniimprimantă a jurnalelor de exploatare, cuprinzînd valori instantanee, valori medii ale variabilelor tehnologice şi indicii economici de funcţionare ;
g) Calculul, afişarea şi înregistrarea valorilor de fiabilitate operativă ;
h) Calculul şi înregistrarea automată a energiei nelivrate, datorită nefuncţionării unor agregate.

Subsistemul DIMEC, este realizat sub forma a trei pupitre-operator modulare şi o consolă pentru operator formată tot din trei module independente ce întregesc schema mnemotehnică a instalaţiei tehnologice.

Consola operatorului permite :

— apelarea imaginilor cu valori actualizate ale mărimilor tehnologice, prin apăsarea cu confirmare a tastelor ce se află distribuite pe schemă ;

— apelarea pe display-uri a tuturor informaţiilor calculate prin apăsarea clapelor funcţionale ale acestora ;

— comanda lămpilor de semnalizare în cazul depăşirii limitelor admise pentru variabilele analogice şi pentru indicatorii de fiabilitate operativă calculaţi.

9.4.3.4. *Structura şi funcţiile subsistemului DIREN*

Subsistemul DIREN cuprinde în structură un microcalculator M118, de asemenea introdus într-un sertar de M18-B pentru a avea o elasticitate mai mare de amplasare a plachetelor, patru module SPOT-80, un display, o miniimprimantă şi o consolă specializată pentru operator.

Din cele patru module SPOT-80, modulul P3 de 64 ieşiri numerice se foloseşte numai pentru comanda elementelor de afişare de pe consola operatorului (fig. 9.32).

Echipamentul este dimensionat pentru a asigura reglarea numerică a 40 variabile tehnologice, după scheme ale reglării convenţionale transpuse în variantă discretizată.

Urmează ca după o perioadă de experimentări şi puneri la punct să se asigure şi programe pentru conducerea procesului cu modele multivariabile.

B	Canal 0	ER 00	Poz 00	L	Canal 0	ER 00	Poz 00	W	Canal 0	ER 00	Poz 00	W_I	Canal 0	ER 00	Poz 00	RN	Canal 0	ER 00	Poz 00
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
5	6	7	8	5	6	7	8	5	6	7	8	5	6	7	8	5	6	7	8
A_B	C	C/M	S	A_L	C	C/M	S	A_W	C	C/M	S	A_I	C	C/M	S	A_N	C	C/M	S

Fig. 9.33. Consola de operator pentru subsistemul de reglare numerică DIREN.

Consola operatorului (fig. 9.33), concepută pentru blocurile de 330 MW, funcționînd pe cărbune, este prevăzută cu cinci blocuri numerice de reglare pentru combustibil (B), aer (L), apă de alimentare (W), injecții (I) și sarcină (N).

Fiecare bloc are posibilitatea reglării independente a opt variabile tehnologice, pentru care se poate afișa pe consolă numărul canalului, eroarea între valoarea variabilei reglate și valoarea de consemn, precum și poziția organului de execuție.

Fiecare bloc de reglare, în afara selectării canalului dorit, mai are posibilitatea de trecere, fără șocuri, de pe „manual“ pe „calculator“ și invers, iar în poziția „manual“ de a comanda fiecare element de execuție la „crește“ și la „scade“.

Acționarea tastelor de comandă ale blocului devine activă numai în cazul în care se apasă și tasta de adresă a acestuia.

Prezentăm, în continuare, un exemplu de utilizare a unui bloc de reglare. Blocul de reglare numerică cu opt canale și indicativul „I“ se folosește pentru reglarea numerică a celor șase temperaturi de abur primar și secundar de la grupul de 330 MW.

La apelul tastei cu indicativul buclei de reglare dorite, în afară de informațiile privitoare la numărul canalului, a erorii de reglare și a poziției organului de execuție, pe display-ul subsistemului DIREN apare automat schema structurală a buclei de reglare, pe care se actualizează la fiecare scanare a parametrilor tehnologici, valoarea temperaturii reglate, valoarea de consemn stabilită, temperatura aburului după injecție, debitul apei de injecție și se repetă poziția organului de execuție.

Pe această imagine (fig. 9.34) mai pot fi notate și alte informații privitoare la bucla de reglare respectivă și utile operatorului (ex. valorile de acordare).

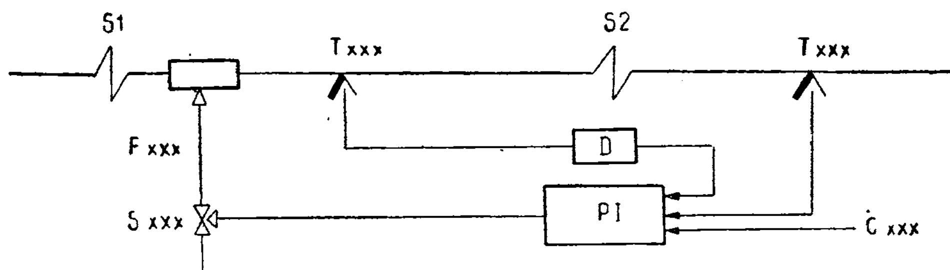


Fig. 9.34. Imaginile display-reglare temperatura abur (exemplu).

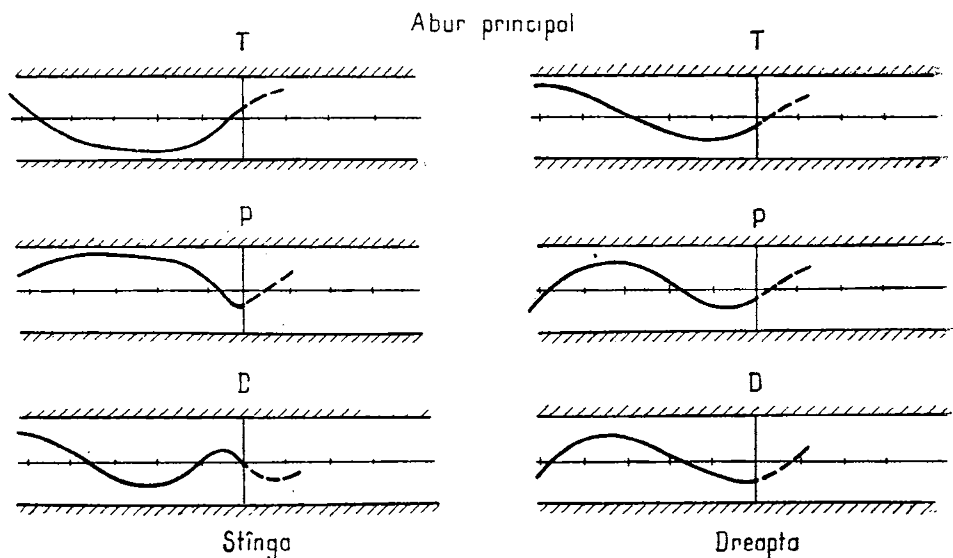


Fig. 9.35. Diagrame de funcţionare-abur principal (exemplu).

Tastele funcţionale ale display-ului se folosesc pentru a afişa *diagramele de funcţionare curente şi de estimare* a principalelor variabile tehnologice reglate.

Cu ajutorul acestor diagrame de funcţionare se poate urmări evoluţia principalilor parametri reglaţi pe o perioadă anterioară de cinci minute şi estima, prin modele predictive, evoluţia în continuare (un minut) a aceloraşi parametri.

Modelul se bazează pe estimarea evoluţiei anterioare cu ajutorul metodei celor mai mici pătrate şi pe predicţia prin serii recursive de timp a evoluţiei viitoare.

Aceste diagrame de funcţionare curentă se prezintă sintetic pentru principalele mărimi tehnologice ce caracterizează funcţionarea instalaţiei şi constituie pentru operator una din imaginile cele mai bogate în informaţii (fig. 9.35).

Predicţia evoluţiei viitoare a parametrilor permite să se facă o presemnalizare anticipînd situaţiile ce ar putea deveni critice.

La un multiplu al perioadei de scanare a variabilelor tehnologice imaginea este reactualizată pe principiul ferestrei mobile.

9.4.3.5. *Arhitectura sistemului SINEX-82*

Un schimb minim de informaţii între cele trei subsisteme, DICREV, DIMEC, DIREN, creează posibilitatea realizării ansamblului de funcţii prevăzute pentru SINEX-82.

Întregul sistem funcţionează cu o bază de timp unică, luată de la unul din subsisteme, care se transmite la celelalte ca un ceas extern.

Din punct de vedere arhitectural echipamentul electronic se montează în dulapuri operative pentru sertare de 19″ în care sînt înglobate elementele de afişare pe tub catodic şi consola operatorului.

Consola sistemului SINEX-82 cuprinde trei module independente, care alăturate întregesc într-o structură mozaicată schema mnemotehnică a instalației tehnologice pe care sînt distribuite tastele de comandă ale subsisemului DIMEC, tastatura consolei de proces a subsistemului DICREV și cele cinci blocuri ale tastaturii subsistemului de reglare numerică DIREN.

Pe fiecare modul al consolei operatorului se află cîte o tastă de confirmare a comenzilor ce se pot da de către operator, fără de care orice altă comandă nu este luată în considerare de sistem.

În varianta standard se folosește o arhitectură cu cinci display-uri și trei dulapuri de dimensiuni reduse comparativ cu restul echipamentului, ca în figura 9.36.

Din cauza dimensiunilor ecranelor de 31 cm, operatorul trebuie să se afle la o distanță de pînă la 2—3 m de pupitru.

Sistemul SINEX-82 poate fi livrat și cu monitoare TEHNOTON, cu diagonala de 44 cm, care asigură o bună definiție și stabilitate a imaginilor.

În acest caz operatorul poate urmări imaginile ecranelor și de la o distanță de 4—5 m, ceea ce îi permite o mai mare mobilitate.

Un alt avantaj important al soluției cu monitoare TEHNOTON este faptul că nu mai este obligatorie amplasarea acestora în apropierea plachetei de comandă VDI, ci se pot transmite imaginile corect, prin cablu coaxial, pînă la distanța de 200 m.

În cazuri speciale, folosind adaptoare de linie, această distanță poate fi mult mărită.

Echipamentul capătă în acest fel posibilități mult mai elastice de amplasare, ceea ce este deosebit de important, mai ales în cazul introdu-

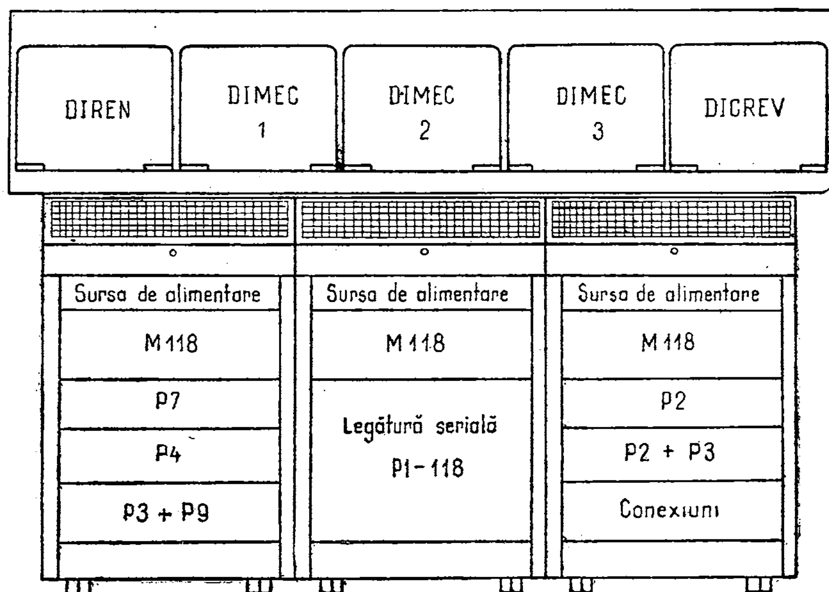


Fig. 9.36. Arhitectura echipamentului SINEX-82.

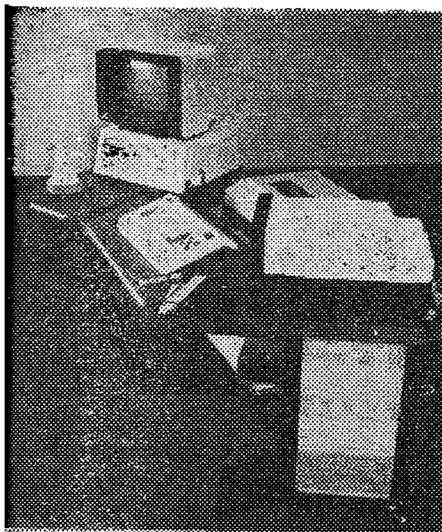
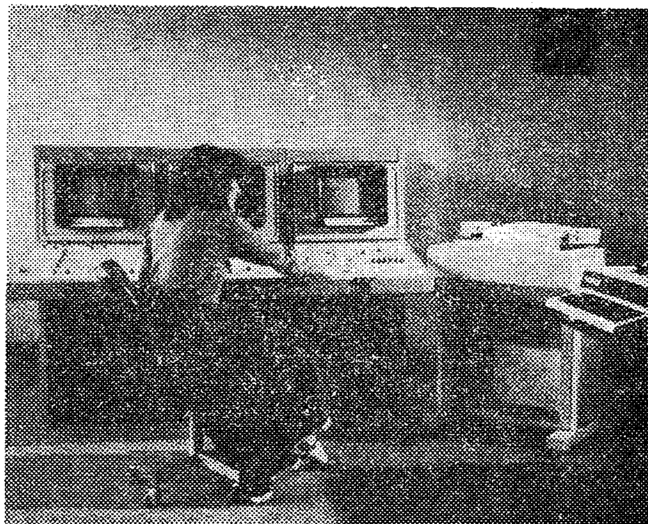



Fig. 9.37. SINEX-82, imagini prototip.

cerii acestuia la centrale cu grupuri în funcţiune, la care soluţia este încadrarea ergonomică în camera de comandă existentă a consolei operatorului şi a display-urilor.

Restul echipamentului de calcul se poate amplasa în funcţie de posibilităţile existente în limita distanţei de pînă la 200 m.

În figura 9.37 se arată două imagini luate în timpul testării prototipurilor, într-o variantă cu trei display-uri şi o zonă a consolei operatorului cu tastele de apel ale imaginilor.

Trebuie menţionat, de asemenea, faptul că în cazul unor aplicaţii specifice subsistemele DICREV, DIMEC sau DIREN, se pot folosi şi independent, renunţînd la unele opţiuni care ţin de comunicaţia dintre acestea.

9.4.4. Organizarea generală a programelor sistemului SINEX-82

Se vor prezenta numai unele elemente introductive legate de sistemul de programe ce asigură funcţionarea echipamentului DIMEC.

Programele de sistem RTX-SINEX asigură :

— actualizarea ceasului de timp real ;

— sincronizarea şi coordonarea taskurilor concurente din sistem ;

— efectuarea calculelor aritmetice în virgulă mobilă ;

— comanda perifericelor ;

— cuplarea a trei monitoare tip dispaly la o singură unitate centrală ;

— cuplarea a trei tastaturi de comandă la o singură unitate centrală ;

— afişarea grafică continuă pe display şi crearea unui set de simboluri grafice specifice proceselor termoenergetice.

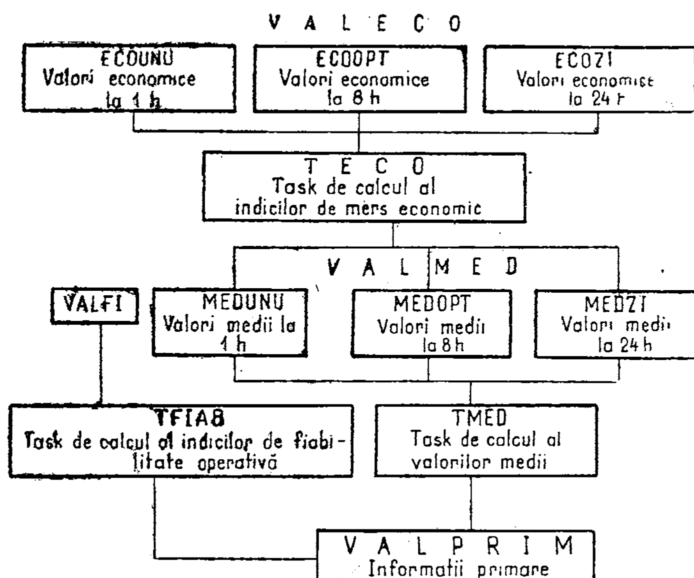


Fig. 9.38. Structura generală-programare de calcul DIMEC.

Programele de aplicație folosesc baza de date furnizată de P1-118, în care se efectuează prelucrările primare (validare, controlul limitelor, filtrarea numerică).

Structura generală a programelor de calcul și a programelor de afișare se poate urmări în figurile 9.38 și 9.39.

Programele sînt astfel structurate, încît pornind de la baza primară de date se creează o nouă bază de date formată din informații de fiabilitate operativă, informații de valori medii, de valori economice și informații ce se transferă la sistemul ierarhic superior — DISCO.

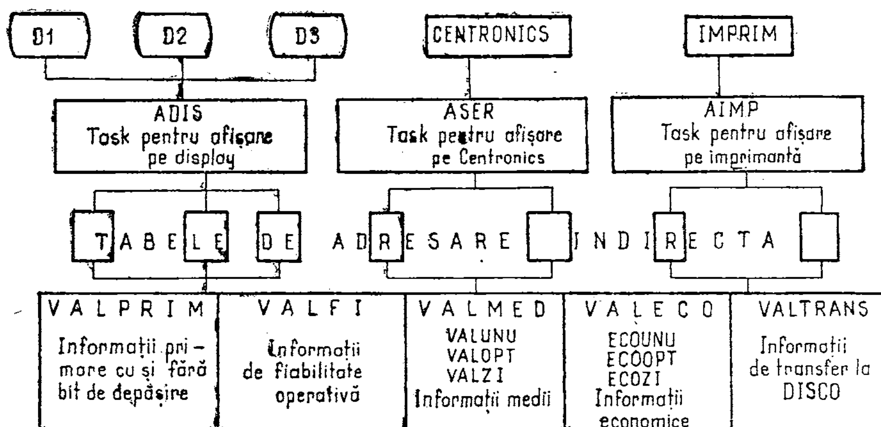


Fig. 9.39. Structura generală-programare de afișare DIMEC.

Noile informații obținute sînt folosite de taskurile de afișare și puse la dispoziția operatorului cu ajutorul perifericelor standard ale sistemului.

Numai pe cele trei display-uri ale subsistemului DIMEC operatorul poate apela 24 imagini cu valori actualizate ale variabilelor tehnologice.

Jurnalele de exploatare se obțin automat de la miniimprimantă, iar pe Centronics sînt înscrise evenimentele ce au loc, printre care, mai ales, ieșirile din limitele prescrise ale parametrilor.

9.5. Aplicarea microcalculatorului M-18 în conducerea procesului de fabricare a celulozei

Industria celulozei și hîrtiei ocupă un loc important în posibilitățile de conducere automatizată a proceselor tehnologice cu ajutorul calculatorului.

Și în această ramură industrială, opțiunea pentru trecerea la introducerea calculatoarelor de proces este justificată de complexitatea instalațiilor, de imposibilitatea menținerii unui regim optim de funcționare ca urmare a variațiilor în calitatea materiilor prime, de obținerea unor caracteristici superioare a produselor finite. În condițiile automatizării convenționale aceste abateri nu pot fi stăpînite decît într-o măsură mai redusă, ca urmare a necuprinderii în ansamblu a influențelor reciproce a variabilelor procesului.

Analiza tehnico-economică a întreprinderilor de fabricare a celulozei și hîrtiei, a întregului ansamblu de operații și procese (fierbere, spălare, evaporare, regenerare, caustizare, măcinare, formarea și consolidarea hîrtiei [10] a fundamentat necesitatea proiectării și implementării sistemelor de conducere cu calculatorul.

Introducerea calculatoarelor de proces în fabricile de celuloză și hîrtie permite mai buna folosire a instalațiilor, reducerea consumurilor specifice și îmbunătățirea calității, astfel :

— creșterea capacității de producție cu cca 6—8% la fierberea discontinuă ;

— creșterea capacității de producție cu cca 10—15% la fierberea continuă ;

— reducerea consumului de reactivi cu cca 5% ;

— scăderea consumurilor specifice de abur și energie electrică.

Primul calculator de proces montat la C.C.H. Suceava, pe o linie de fabricare a hîrtiei sulfat, a atins nivelele planificate ale indicatorilor tehnico-economici în conformitate cu studiul tehnico-economic.

Din gama de echipamente de calcul produse în țara noastră, prin caracteristicile hard și posibilitățile de programare, microcalculatorul M-18 satisface cerințele funcționale și structurale, cu caracter general, impuse de [11] :

— comunicarea direct cu procesul ;

— a răspunde în timp util modificării variabilelor care definesc starea procesului la un moment dat ;

— a executa programe cu nivele de priorități diferite ;

— existența unor programe specifice procesului ;

— posibilitatea realizării unor configurații modulare, impuse de cerințele instalației ;

— posibilitatea interconectării cu alte echipamente de calcul ;

— fiabilitatea înaltă.

Natura și complexitatea procesului tehnologic de fabricare a celulozei sulfat a impus adoptarea uneia din variantele de conducere cu microcalculatoare M-18, îndeplinind una din următoarele funcțiuni [12, 13] :

— Conducerea procesului tehnologic cu calculator de tip „consultant", situație în care calculatorul nu este cuplat direct pe proces, operatorul tehnologic, pe baza unor date introduse spre prelucrare pe baza unui model matematic, a simulării procesului, ia decizii corecte și optime ;

— Conducerea centralizată a procesului tehnologic în regim de supraveghere.

Prin intermediul unor aparate traductoare, ce măsoară anumiți parametri, calculatorul se cuplează pe proces, măsurînd și înregistrînd, compară valorile acestora în vederea emiterii de semnale de avertizare a operatorului în cazuri de depășiri ale valorilor prestabilite și de rapoarte de stare a procesului.

— Conducerea procesului tehnologic cu calculator în regim de „ghid" operator cu sarcini de exploatare și afișare a mărimilor din proces și emiterea de mesaje cu instrucțiuni de operare corespunzătoare diferitelor modificări ale regimului de funcționare ;

— Conducerea procesului tehnologic cu calculator în regim de supervizor, caz în care, calculatorul acționează direct, fără intervenția operatorului, asupra mărimilor procesului pentru comanda elementelor de execuție ce influențează direct conducerea procesului.

În această variantă calculatorul îndeplinește o sarcină suplimentară — optimizarea ; se pornește de la identificarea stării variabilelor și în funcție de aceasta determină valorile optime la care trebuie menținuți parametrii în conducerea procesului.

Conducerea cu calculatorul a procesului tehnologic de fabricare a celulozei presupune în afara analizei de proces, a elaborării concepției globale și a proiectului de detaliu, a execuției echipamentelor și instalării acestora, construcția modelului matematic de conducere.

În elaborarea acestuia, așa după cum vom vedea, se apelează la legi fizice și cinetici de reacție pentru a cuprinde cît mai exact influențele variabilelor procesului. Această etapă se constituie ca un moment deosebit de important în proiectarea sistemului de conducere.

Menținerea unui grad de dezincrustare a lemnului la variații cît mai mici este determinată și de compoziția leșiei de fierbere, în special raportul alcalii active față de lemn, de sulfiditate și de hidromodul.

În practica curentă intervin unele variații aleatoare care influențează gradul de dezincrustare a celulozei și respectiv randamentul și calitatea acesteia.

Variațiile în compoziția leșiei de fierbere, a umidității și structurii dimensionale a tocăturii, determină schimbarea raportului alcalii efective/lemn și în consecință, prin modificarea vitezei de reacție se modifică conținutul de lignină din celuloză.

Pentru caracterizarea cineticii procesului de fierbere se folosește noțiunea de factor H, conform relației :

$$H = \int_0^t e^{\left(43{,}2 - \frac{16\,113}{T}\right)}\ dt \qquad (1)$$

t = timp de fierbere
și care reprezintă suprafața cuprinsă între curba K=(T) și axa timpului.

9.5.1. Unele aspecte ale conducerii optime a procesului de fabricare a celulozei

Procesul de fabricare a celulozei sulfat este recunoscut ca avînd o mare complexitate datorită reacțiilor de dezincrustare a lemnului și a cineticii care nu pot fi controlate în timpul fierberii. O cauză care contribuie la modificări nesupravegheate în calitatea și cantitatea celulozei o reprezintă variațiile aleatoare ale calității lemnului.

Randamentul celulozei sulfat rezultat la fierbere se apreciază prin conținutul de lignină reziduală evaluat cu ajutorul metodei de determinare a gradului de dezincrustare (cifra Kung, Tappi etc.).

Studiile noastre au fundamentat posibilitatea folosirii factorului τ în conducerea procesului de fabricare a celulozei, ca urmare a cuprinderii unui număr de 5 variabile de bază ce definesc suficient de bine cinetica reacțiilor de dezincrustare.

Admițîndu-se că viteza de consum a ionilor de hidroxil este proporțională cu viteza de delignificare, iar concentrația ionilor de sulfhidrat nu se modifică sensibil în timp, durata de fierbere din proces este proporțională cu factorul τ definit ca :

$$\tau = C\left(\frac{Ae}{h}\right)^2 \quad H \qquad (2)$$

unde C este constanta vitezei de reacție, Ae/h raportul dintre alcaliile efective și hidromodul.

Totodată, s-au elaborat modelele matematice ce descriu dependența gradului de dezincrustare, a randamentului și a selectivității fierberii (exprimată prin raportul hidrați de carbon/lignină) de factorul τ, în funcție de variația dimensională a tocăturii.

Instabilitatea unor parametri ai procesului de fabricare a celulozei sulfat a impus construirea unui model matematic autoadaptiv, pornind de la funcția analitică a procesului, de forma [15] :

$$K = F\left(a,\ b,\ c\ \frac{\tau}{100}\right) \qquad (3)$$

și folosind ca criteriu de comparație funcția distanță, definită prin relația :

$$D\left(a,\ b,\ c\ \frac{\tau}{100}\right) = \left|\ Fc\left(a,\ b,\ c\ \frac{\tau}{100}\right) - Fm\left(a,\ b,\ c\ \frac{\tau}{100}\right)\ \right| \qquad (4)$$

Dacă $D<\varepsilon$ cu ε (grad de adecvanță) bine precizat și cu valori mici, atunci algoritmul de ajustare nu mai are loc. Algoritmul de ajustare se aplică pentru $D>\varepsilon$ și funcționează pe baza principiului minimizării unei funcționale. Pentru minimizare s-a ales o metodă de căutare a minimului fără evaluarea derivatei, pe mai multe direcții (metoda Powell).

Analiza procesului de fabricare a celulozei sulfat — procedeu discontinuu — cercetarea în condiții de laborator și verificarea în condiții industriale a unor noi soluții de perfecționare a tehnologiei de fabricație s-au constituit în etape care au fundament posibilitățile de îmbunătățire a parametrilor de eficiență prin folosirea unui calculator de proces.

9.5.2. Conducerea procesului de fabricare a celulozei cu microcalculatorul M18 în regim de „consultant"

Din considerente ce țin de siguranța în funcționare a instalațiilor de fierbere a celulozei, cît și din lipsa posibilităților de a se realiza o interfață proces-calculator, la acea dată, s-a ales varianta conducerii asistate de calculator a acestui proces.

În fig. 9.40 se prezintă fluxul de date și informații în sistemul de conducere asistată de calculator a fierberii sulfat.

În sistemul de conducere asistată de calculator a fierberii sulfat — procedeu discontinuu — operatorul panoului tehnologic transferă microcalculatorului date referitoare la variabilele procesului (compoziția leșiei de fierbere, cantitatea de lemn) și recepționează valorile pentru regimul tehnologic ce se impune de obținerea unui anumit grad de dezincrustare sau de randament.

Pe baza modelelor matematice existente în memoria microcalculatorului, se calculează volumul de leșie albă și neagră la un anumit raport de alcalii efective/lemn, timpul de menținere în palier pentru un anumit factor H calculat, la o valoare prestabilită a temperaturii finale și a timpului de urcare.

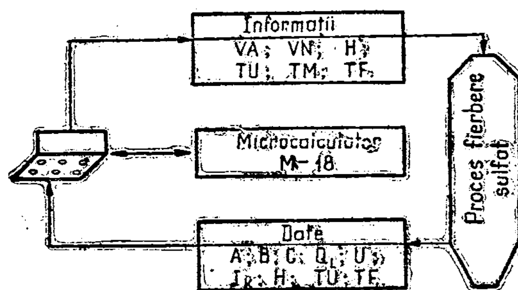


Fig. 9.40. Fluxul de date și informații în sistemul de conducere asistată de calculator a fierberii sulfat:

- A, B, C — conținutul în NaOH, Na_2CO_3, Na_2S a leșiei de fierbere;
- VA — volumul de leșie albă introdus în fierbător — m^3;
- VN — volumul de leșie neagră necesar pentru realizarea hidromodulului prescris — m^3;
- Q_L — cantitatea de tocătură umedă, t_0;
- U — umiditatea lemnului, %;
- I_K — indice Kung;
- TU — timp de urcare în palier, min;
- TM — timp de menținere în palier, min;
- TF — temperatura finală de fierbere, °C;
- H — factor;
- S_1, S_2 — subrutine de lucru pentru introducerea datelor în vederea obținerii regimului optim de fabricație.

Cu ajutorul microcalculatorului M18 se realizează următoarele funcții în conducerea procesului :

a) controlul încărcării cu tocătură de lemn ;

b) controlul și dozarea cantității de alcalii efective ;

c) calculul regimului de fierbere a lemnului (temperatură și timp de fierbere) ;

d) determinarea noilor parametri ai fierberii în cazul unor incidente independente de operator pentru atingerea gradului de dezincrustare dorit ;

e) calculul randamentului în celuloză obținut pe baza gradului de dezincrustare și respectiv a producției de celuloză.

SOFTWARE-ul aplicației este format dintr-un pachet de programe, organizat într-o structură arborescentă, la care se distinge (fig. 9.41) :

— un program director ce asigură inițializarea lucrului între operator și calculator, accesul la segmentele program specializate pe anumite operații ;

— un număr de 7 segmente de program cu funcțiuni specifice.

Se execută următoarele operații de conducere a fierberii sulfat :

— calculul compoziției leșiei de fierbere (S_{11}) ;

— calculul proporțiilor (AE/lemn și hidromodul (S_{12}) ;

— calculul factorului H pentru un grad de dezincrustare impus (S_{13}) ;

— calculul cantității de leșie albă la un procent dat de AE/lemn (S_0) ;

— determinarea timpului de menținere în palier la o valoare prestabilită a factorului H (S_2) ;

— calculul de optimizare a procesului atunci cînd se ia în considerare structura dimensională a tocăturii (SL, SL_1, SL_2, SL_3) ;

— calculul randamentului în celuloză în funcție de cantitatea de alcalii, timp fierbere și temperatură (SR) ;

— calculul temperaturii de fierbere în cazul unor abateri de la regimul tehnologic prescris (ST) ;

— corecția parametrilor modelului (SC).

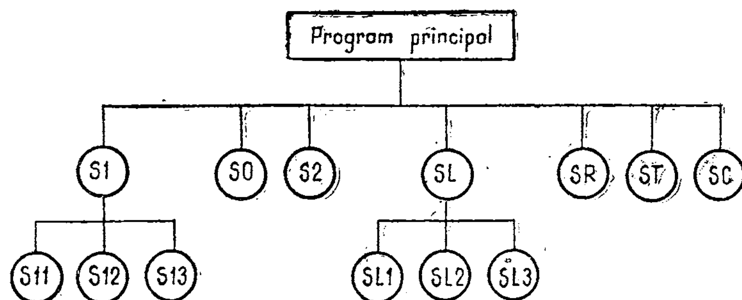


Fig. 9.41. Software-ul aplicativ privind conducerea asistată de calculator a fierberii sulfat :

S_1, S_{11}, S_{12}, S_{13}, S_0, S_2, S L, SR, ST, SC — subrutine specializate în cadrul pachetului de programe de conducere asistată de calculator a fierberii sulfat.

Configurația sistemului este următoarea :
- unitate centrală și memorie de 48 KO ;
- unitate de bandă magnetică ;
- 2 unități de casete magnetice ;
- lector de cartele ;
- lector de bandă perforată ;
- imprimantă.

9.5.3. Elaborarea cu ajutorul microcalculatorului M18 a bilanțului de fibră și săruri sodice la instalația Kamyr de fabricare a celulozei

O aplicație deosebit de importantă pentru conducerea fierberii în regim continuu pe o instalație de fabricare a celulozei de tip Kamyr o reprezintă elaborarea bilanțului de materiale (7).

Lucrul interactiv al tehnologului cu microcalculatorul M18 asigură calcularea operativă și cunoașterea următoarelor consumuri :
- consumul de lemn/t celuloză ;
- randamentul termic al fierbătorului ;
- consumul de alcalii active ;
- consumul de leșie neagră.

Se determină de asemenea :
- necesarul de leșie neagră și hidromodulul ;
- substanța uscată în leșia neagră ;
- concentrația leșiei negre la sfîrșitul fierberii ;
- substanțele solubile recuperabile ;
- pierderile de substanță uscată la filtrul 1 ;
- apa eliminată la expandare etc.

Modelele matematice de calcul a parametrilor instalației Kamyr dă posibilitatea simulării în ansamblu a procesului de fabricare a celulozei sulfat. Orice tehnolog are la îndemînă o strategie de abordare a problemelor de analiză a procesului de fabricare a celulozei sulfat. Modificînd în modelele matematice valorile diferitelor variabile se obține imediat răspunsul la aceste schimbări, scurtînd prin aceasta perioada de experimentări industriale și chiar înlăturarea unor pierderi ce pot apare în asemenea cazuri.

Bilanțul instalației Kamyr ia în considerare un număr mare de parametri încît simularea procesului de fierbere, privit ca un ansamblu unitar, conduce la determinarea performanțelor tehnico-economice în diverse variante de lucru.

În figura 9.42 se prezintă schematic structura sistemului de programe a bilanțului instalației Kamyr, în vederea analizei și operării în condiții optime în fabricarea celulozei sulfat.

Simularea instalației Kamyr prin utilizarea microcalculatorului M-18 se înscrie în cadrul general ce permite obținerea următoarelor avantaje [15] :

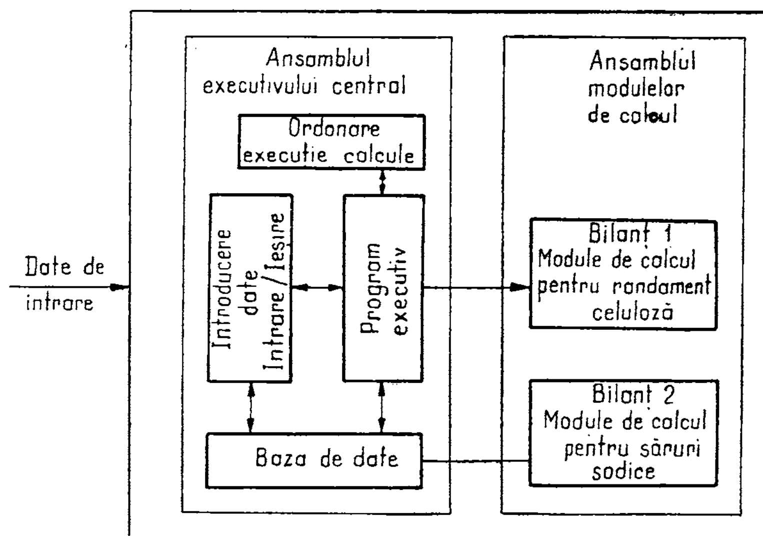


Fig. 9.42. Sistemul de analiză şi elaborare a bilanţului Kamyr.

— studierea în timp scurt a unui număr mare de variante de dimensionare şi operare ;

— analiza funcţionării instalaţiei de fierbere a celulozei în vederea îmbunătăţirii consumurilor specifice şi a randamentului în celuloză ;

— evitarea unor faze de cercetare ce nu se pot realiza la scară industrială sau nu prezintă rezultate concludente în instalaţii pilot.

9.5.4. Conducerea supravegheată a procesului de fabricare a celulozei cu microcalculatorul M-18

Prin Institutul Politehnic Bucureşti s-a conceput şi executat o interfaţă M-18 — proces pentru conducerea supravegheată a fierberii sulfat.

În realizarea interfeţei s-a avut în vedere :

— achiziţia de mărimi analogice din proces ;

— coordonarea simultană a patru fierbătoare ;

— un interval mediu de eşantionare de cca 1 minut.

Particularităţile procesului de fabricare a celulozei sulfat şi conducerea optimală a acestuia avînd la bază urmărirea regimului de temperatură, precum şi caracteristicile hardware ale microcalculatorului M-18, au impus realizarea unei interfeţe specializate (fig. 9.43).

Pentru achiziţii de date s-a folosit un convertor numeric-analog de 8 biţi şi un comparator, conversia analogic-numeric făcîndu-se prin program, iar multiplexarea intrărilor din proces cu relee Reed.

Conducerea în timp real a procesului s-a făcut printr-un ansamblu de programe care să cuprindă îndeplinirea funcţiunilor prezentate în capitolul anterior. În utilizarea limbajelor de programare s-a folosit o soluţie de compromis prin utilizarea limbajului FORTRAN pentru subruti-

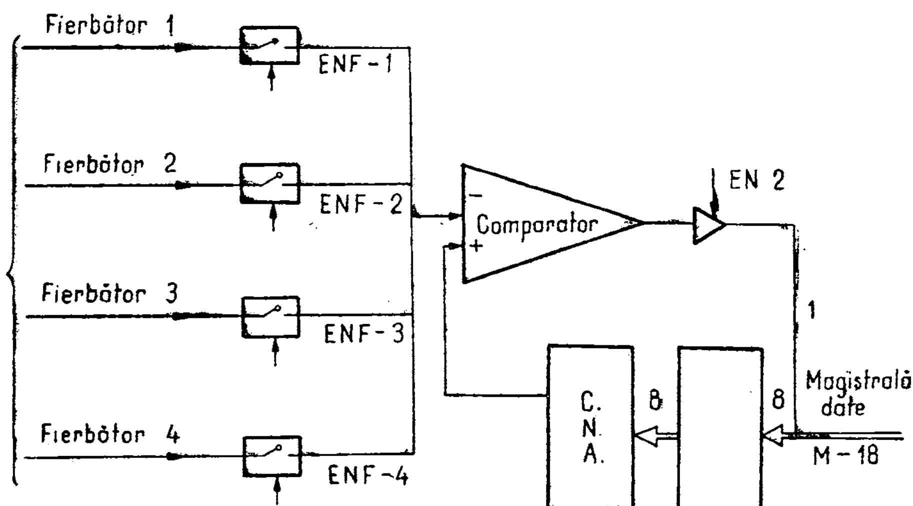


Fig. 9.43. Interfața pentru achiziție de date.

nele de conducere, iar pentru rutina de întrerupere de timp real, limbajul de asamblare.

Programul este de tip monitor, acceptă comenzi specifice de la consolă, care permit [18] :

— inițializarea unei fierberi (calculul parametrilor fierberii) ;

— afișarea la cerere a situației de moment a regimului de lucru a fierbătorului (temperatură, factor H etc.) ;

— executarea unor corecții în cazul unor abateri fortuite de la regimul prescris ;

— semnalizarea terminării fierberii și oprirea mesajului de alarmă.

Conducerea supravegheată a fabricării celulozei sulfat prin conectarea unui microcalculator M-18 pentru calculul regimului dorit de fierbere, achiziția de date și supravegherea temperaturii ca variabilă de comandă, a avut ca obiective optimizarea producției prin :

— alegerea gradului de dezincrustare dorit al celulozei cu implicații directe asupra randamentului și calității acesteia ;

— sporirea capacității instalației de fierbere prin scurtarea timpului de fierbere, ca urmare a determinării cu exactitate a tuturor influențelor aleatoare induse de calitatea lemnului și a agenților dinamici de fierbere ;

— reducerea consumurilor specifice de lemn și săruri sodice.

În conceptul conducerii optimale s-au folosit modele matematice autoadaptive, care, în contextul schimbărilor caracteristicilor materiilor prime, execută corecția coeficienților variabilelor modelului.

În felul acesta gradul de dezincrustare calculat se acordează permanent cu gradul de dezincrustare rezultat din proces.

Schema de ansamblu de conducere optimă în regim de supraveghere a celulozei sulfat este dată în figura 9.44, [19].

Efectele economice ce se obțin ca urmare a conducerii cu microcalculatorul M-18 a fierberii sulfat a celulozei sînt deosebit de importante, prin [20] :

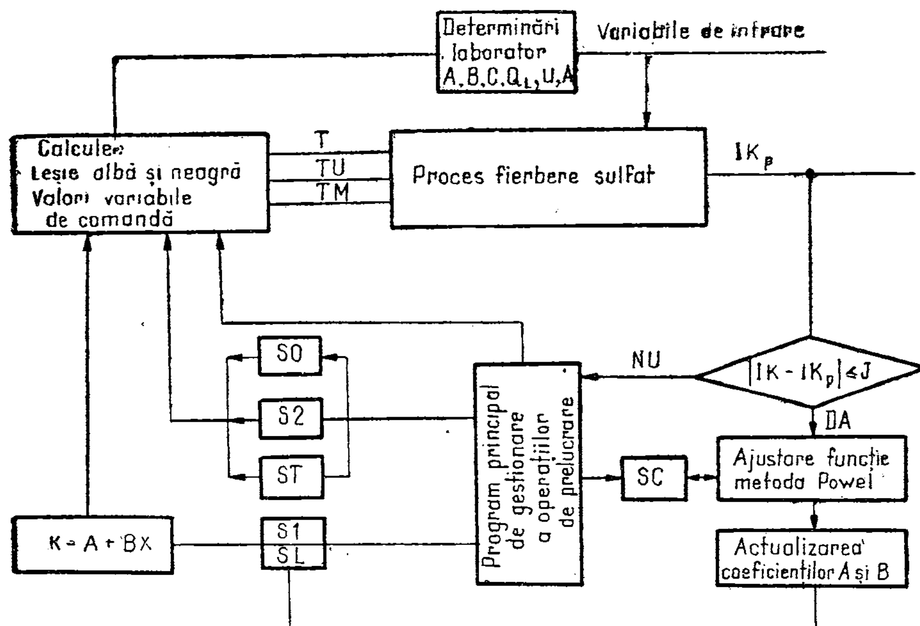


Fig. 9.44. Schema sistemului de conducere optimală a procesului de fierbere a celulozei sulfat.

— creșterea capacității de producție de pînă la 6% ;
— creșterea randamentului în celuloză cu 0,5% ;
— reducerea consumurilor specifice de lemn și săruri sodice ;
— îmbunătățirea caracteristicilor de calitate etc.

9.6. Aplicarea microcalculatorului M-18 la conducerea procesului de preparare a minereurilor neferoase

9.6.1. Analiza procesului de preparare

Industria minieră din țara noastră este pusă, în contextul penuriei actuale de combustibil și materii prime, în fața unor cerințe de ordin cantitativ și calitativ.

Tendințele în acest domeniu sînt determinate de :

— creșterea în dimensiuni, capacitate și complexitate a instalațiilor tehnologice industriale ;

— obținerea unor caracteristici superioare a calității produselor, a productivității și randamentelor ;

— îmbunătățirea gradului de siguranță în funcționarea instalațiilor ;

— restricții impuse de economia de combustibil și materie primă, de protecția mediului înconjurător.

În fața acestor imperative conducerea proceselor de preparare a minereurilor trebuie să adopte strategii noi care să permită :

— un regim stabil și optim al procesului tehnologic din punct de vedere economic în condițiile unor variații mari de conținuturi în metale a minereurilor și a creșterii dificultăților de conducere de către operator a tehnologiei de fabricație ;

— acționarea de la distanță asupra unor elemente de execuție din instalația tehnologică ;

— modificări și perfecționări ale procesului tehnologic prin intermediul unor echipamente de automatizare flexibile ;

— culegerea, transmiterea, memorarea și prelucrarea unui volum mare de date ;

— optimizarea procesului tehnologic pe baza modelării matematice a comportării statice și dinamice a acestuia.

Soluția din punct de vedere tehnic în realizarea obiectivelor de mai sus ține de introducerea și utilizarea calculatoarelor de proces.

Realizările pe plan mondial în reglarea și optimizarea instalațiilor de flotație au arătat că acestea sînt pretabile la conducerea cu calculatorul.

O condiție maximală pentru conducerea automată a procesului cu calculator este măsurarea în flux a conținutului de metale utile.

Analiza comparativă a diverselor sisteme de conducere a unor flotații similare cu cea de la Exploatarea Tarnița evidențiază următoarele posibilități în conducerea procesului :

— reglajul reactivilor de tip colectori pentru a menține punctul de funcționare a procesului la nivelul superior al gradului de recuperare ;

— reglajul cantității de apă din tulbureală pentru a optimiza funcționarea uzinei din punct de vedere al calității concentratului funcție de gradul de recuperare ;

— reglajul flotabilității minereului în sensul diminuării efectelor negative ale modificărilor tipului de minereu.

Preparația de minereuri neferoase Tarnița prelucrează următoarele tipuri de minereu :

— minereu cuprifer (minereu cuprifer compact și de impregnație) ;

— minereu complex, cu obținerea în urma concentrării a următoarelor produse :

- concentrat cupros ;
- concentrat piritos ;
- concentrat plumbo-zincos.

În cadrul fluxului tehnologic se disting operații specifice de : antezdrobire, măcinare, flotare, îngroșare, ce necesită un control permanent asupra parametrilor, prin aparatură de măsură și control adecvată.

Principalii parametri ai procesului de flotație sînt :

- puterile electrice consumate de mori la concasare ;
- încărcarea benzilor transportoare ;
- debite de apă la mori ;
- debite de reactivi ;
- densitate tulbureală la intrare în celulele de flotație ;
- conținutul de metale în minereu și în tulbureală ;
- *p*H-ul tulburelii etc.

9.6.2. Proiectarea și implementarea conducerii cu calculator

Proiectarea și implementarea unui sistem de conducere cu calculatorul a flotației de minereuri neferoase Tarnița a implicat abordarea unor activități complet distincte și care vizau :

— modernizarea tehnologiei de preparație ;

— perfecționarea automatizării întregii instalații ;

— introducerea calculatorului de proces.

Activitatea de proiectare și implementare a microcalculatorului M-18 pentru flotația Tarnița s-a desfășurat în paralel cu experimentarea și aplicarea măsurării discontinue sau continue a conținuturilor de metale din minereu și tulbureală, cu proiectarea și montarea aparaturii convenționale de măsură și control.

În acest ansamblu de măsuri, utilizarea metodelor matematice de analiză a datelor, de modelare și optimizare, au reprezentat instrumente de cercetare și proiectare asistată de calculator în toate etapele de realizare.

Evoluția conducerii cu calculatorul a procesului de flotație a minereurilor neferoase este similară cu cea a fabricării celulozei sulfat prezentată anterior, cuprinzînd ca etape distincte :

— analiza și modelarea procesului de flotație a minereurilor neferoase ;

— conducerea asistată de calculator (FELIX C-256) a procesului de flotație ;

— conducerea on-line a flotației.

De aceea, ne vom ocupa în continuare de problematica conducerii on-line a preparației de minereuri neferoase.

În perioada premergătoare introducerii microcalculatorului M-18 pentru supravegherea și conducerea procesului, pe baza analizei statistice a datelor experimentale din proces s-a concluzionat că flotația minereului neferos din această uzină prezintă o mare instabilitate în reglajul reactivilor, în special a xantatului, cu tendințe de :

— adiție excesivă de xantat și deci posibilități de reducere a consumului ;

— influență negativă asupra procesului în ansamblu.

În același mod s-au putut determina influențele variabilelor procesului asupra gradului de recuperare a metalelor utile din minereu și s-au fundamentat obiectivele ce trebuiesc avute în vedere în etapa finală de conducere cu calculatorul.

Față de situația existentă, conducerea cu calculatorul vizează ca obiective :

— obținerea unei cantități maxime de concentrat prin utilizarea la maxim a capacităților de producție ;

— îmbunătățirea calității concentratului prin dozajul corespunzător al reactivilor ;

— reducerea consumurilor specifice de reactivi, materiale, energie etc.

În figura 9.45 se prezintă sistemul calculator — proces pentru conducerea procesului de flotație la E.P. Tarnița.

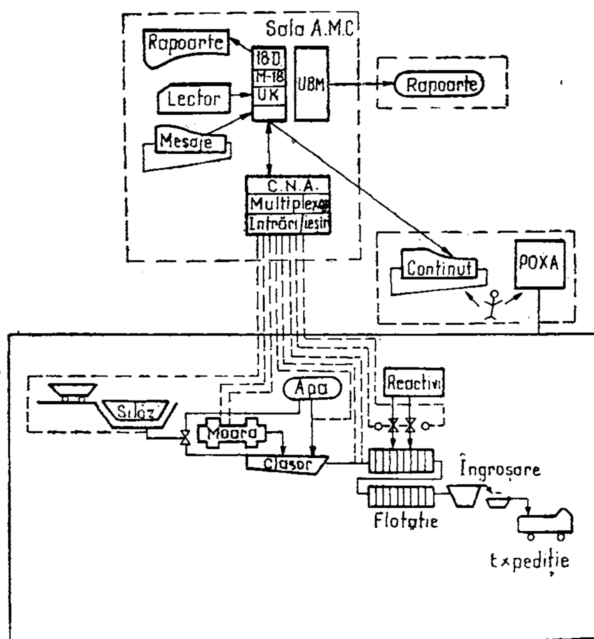


Fig. 9.45. Sistemul proces-calculator folosit pentru conducerea procesului de flotaţie.

Configuraţia hard este următoarea :

1. Microcalculator M-18 cu :
   - unitate centrală de 58 KO ;
   - unitate de bandă magnetică ;
   - unitate de casetă ;
   - unitate de lector de bandă perforată ;
   - imprimantă (132 caractere pe linie) ;
   - consolă CENTRONICS ;
   - lector de cartele ;
   - terminal CENTRONICS ;
   - DAF 1001.
2. Interfaţa proces-calculator (I.P. Bucureşti), cu următoarele caracteristici :
   - 8 intrări numerice ;
   - 6 intrări analogice (6—10 V) ;
   - 2 ieşiri analogice (6—10 V) ;
   - 1 interfaţă serială USART.

În această componenţă hard s-a avut în vedere reglarea xantatului şi a sulfatului de cupru ca variabile principale de conducere.

Funcţiunile microcalculatorului M-18 de conducere a procesului de preparaţie a minereurilor neferoase sînt :

1. Culegerea de date din proces cu următoarele secvenţe :
   — selectarea punctelor din proces ce urmează a fi culese ;

— determinarea adresei fizice a parametrului în cadrul interfeței specializate de conectare a calculatorului la proces ;
— comanda dispozitivului de multiplexare ;
— filtrare, condiționare semnal ;
— conversia analog numerică ;
— stocare de date în memoria calculatorului.

2. Tratarea situațiilor de excepție din proces.
Situațiile de excepție din proces ce pot apare sînt :
— defectarea traductoarelor de măsură ;
— defectare elemente de execuție ;
— oprirea unor utilaje (mori autogene) etc.

3. Prelucrarea datelor din proces.

În această fază se prelucrează, în afara celor prezentate în cadrul culegerii de date, unele condiții de alarmare și protocolare, modele de reglare etc.

Stoftware-ul specific aplicației are la bază sistemul de operare al microcalculatorului M-18 (MON-18), un sistem de operare în timp real (EPTAR), biblioteca aritmetică FPAL și limbajele de programare PLM/80, ASM80, FORTRAN IV.

Fluxul general de prelucrare a datelor este dat în figura 9.46 și este elaborat pe baza unui executiv de lucru în timp real ce asigură execuția concurentă a mai multor task-uri (programe) pe M-18.

Caracteristica de timp real a sistemului creat semnifică proprietatea sistemului de a asigura răspunsul în timp util la stimuli externi, de exemplu acționarea unor taste de la terminale de către utilizatori, iar prin capacitatea de a asigura rularea concurentă a mai multor programe, sistemul putînd deservi practic mai mulți utilizatori independenți.

Construcția interfeței proces-calculator și implicit programele de culegere date s-au făcut cu timpi de exploatare a variabilelor relativ mari, 1 la 20 sec., deoarece procesul de flotație este un proces lent.

Sistemul de operare este suficient de flexibil, încît pentru toate variabilele cuplate la calculator timpul de exploatare este parametrizat, parțial putînd fi modificat după necesitate.

După cum s-a menționat, interfața proces-calculator gestionează 6 intrări analogice (densitate tulbureală, debit tulbureală, debit apă moară, debit apă clasor, alimentare minereu, *p*H tulbureală) și 8 intrări nume-

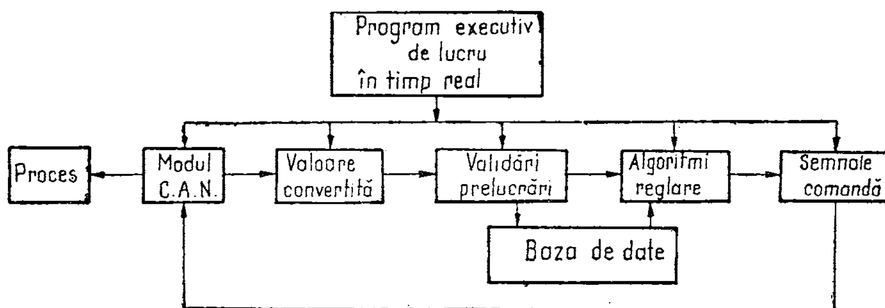


Fig. 9.46. Fluxul general de prelucrare a datelor.

rice de tip contact pentru semnalizarea următoarelor situații: buclă xantat „pe manual“, buclă $CuSO_4$ „pe manual“, moară autogenă oprită, moară autogenă alimentează două linii, robinet xantat complet închis, robinet $CuSO_4$ complet deschis, robinet $CuSO_4$ complet închis.

Citirea intrărilor analogice și numerice se face prin programarea unor porturi ale M-18, iar multiplexarea acestora prin relee REED.

Aplicația de conducere a unei linii de flotație cu microcalculatorul M-18, în configurația prezentată mai sus, este în funcțiune de cca 1,5 ani și se dezvoltă în paralel cu construcția, testarea și introducerea analizoarelor de măsurare a conținuturilor de metale din tulbureală și a aparaturii de măsură și control.

În același timp, pentru uzina de preparație se proiectează o aplicație de urmărire operativă a producției, tot pe un microcalculator M-18, cu următoarele obiective:

— realizarea la nivel de schimb a unei evidențe a principalelor materii prime utilizate în procesul de flotație (minereu, reactivi);

— urmărirea producției realizate pe sortimente la nivel de schimb;

— elaborarea zilnică a rapoartelor de producție;

— evidența calității producției finite.

Culegerea și prelucrarea datelor se face în timp real prin intermediul unor terminale instalate în puncte de control și decizie și a microcalculatorului M-18. Sistemul de urmărire în timp real a producției execută și bilanțul tehnologic de metale la nivel de zi, semidecadă, decadă și lună.

În etapa finală, cele două microcalculatoare M-18 vor fi cuplate între ele, eliminîndu-se unele operații de introducere de date de la terminale. Sistemul de conducere cu microcalculatorul M-18 a procesului tehnologic de flotație a minereurilor neferoase și de urmărire operativă a producției se constituie ca un concept unitar al unui sistem informatic industrial.

Aplicațiile ce au fost prezentate anterior au necesitat eforturi de realizare a unei aparaturi specifice sistemelor automate, ca: interfețe proces-calculator, tastaturi funcționale, panouri de afișaj etc.

În domeniul informaticii industriale, microcalculatorul M-18 va sta la baza proiectării și implementării sistemelor de conducere automatizată a proceselor tehnologice prin dezvoltarea următoarelor componente:

— produse informatice generalizabile pentru culegerea, prelucrarea și transmiterea datelor din proces, identificarea proceselor industriale, algoritmi și metode de optimizare etc.;

— concepția și realizarea de interfețe proces-calculator într-o gamă diversificată și modulară;

— panouri operator și panouri de afișare locală a informațiilor despre proces;

— controlere industriale ca nivel intermediar de automatizare și integrarea acestora cu microcalculatoare M-18 în sisteme ierarhizate de conducere a proceselor industriale.

9.6.3. Concluzii

În raport cu calculatoarele mari, microcalculatorul M-18 cîştigă din ce în ce mai mult teren prin varietatea posibilităţilor de utilizare şi ca urmare a :

— configuraţiei „hard" modulare ce poate satisface o gamă largă de aplicaţii, din diverse domenii de activitate ;

— nivelului tehnic de realizare pentru a răspunde cerinţelor de exploatare cu un înalt grad de fiabilitate ;

— posibilităţilor de programare în unele limbaje evoluate ;

— obţinerii unor performanţe ale sistemelor automate de conducere şi de eficienţă economică în conducerea proceselor industriale ;

— nivelului de dezvoltare a informaticii în ţara noastră ce asigură punerea în funcţiune de aplicaţii în perioade relativ mici de la montare ;

— compatibilităţii cu calculatoarele din familia FELIX.

Concepţia şi realizarea microcalculatorului M-18 în deplină concordanţă cu cerinţele de dezvoltare ale economiei naţionale, reprezintă un element novator în conducerea proceselor industriale.

9.7. Utilizarea microcalculatorului Felix M-18 în sisteme de culegere a datelor şi pontaj automat

Aplicaţia de culegere a datelor se constituie ca o componentă importantă în structura sistemului informatic de întreprindere, deoarece realizează transferul datelor şi informaţiilor direct de la surse, la unităţile de prelucrare şi în continuare la elementele de decizie. Trecerea de la metodele tradiţionale de culegere şi transmitere a datelor (formulare, bonuri, note etc.) la sistemul automat a fost impusă de modernizarea sistemelor de prelucrare şi stocare a datelor prin introducerea calculatoarelor şi a sistemelor de gestiune a bazelor de date.

Funcţionarea şi utilizarea eficientă a echipamentelor sistemului informatic de întreprindere impun modificarea procedurilor clasice de culegere a datelor şi dezvoltarea unor noi tipuri de echipamente şi programe care să satisfacă cerinţele de automatizare în acest domeniu.

Astfel, suporturile de date existente în prezent nu pot fi utilizate pentru introducerea datelor direct în calculator, deoarece echipamentele de intrare nu au capacitatea să preia datele reprezentate în forma respectivă. Încercări de a realiza echipamente de intrare care să citească direct documentele scrise de mînă sau de maşină au condus la obţinerea unor dispozitive sofisticate şi scumpe în raport cu cerinţele de culegere automată a datelor din întreprinderile economice. Concluzia este valabilă şi pentru echipamentele care admit introducerea datelor exprimate prin voce.

Sistemele de culegere automată reduc volumul de date care trebuie înscrise în formulare şi preiau cea mai mare parte a datelor şi informaţiilor direct de la sursă prin intermediul unor terminale specializate.

Prin urmare, sistemele de culegere a datelor, realizează în cadrul sistemelor informatice, mediul de transmisie automată între date, pe de o parte și echipamentele de prelucrare și stocare a datelor (calculatoare și baze de date) pe de altă parte.

Sistemele de culegere a datelor, trebuie să îndeplinească o serie de cerințe, pentru a putea fi mai eficiente și mai performante decît sistemul manual utilizat în prezent.

În primul rînd, timpul scurs între momentul generării datelor și informațiilor și momentul în care sînt disponibile pentru a fi preluate este incomparabil mai scurt în cazul sistemelor automate de culegere. Se spune că datele sînt culese în timp real.

În al doilea rînd, sistemul automat de culegere trebuie să asigure o preluare a datelor eliminînd cît mai mult posibilitatea erorilor. Aceasta presupune ca programele de culegere să fie completate în activitatea lor de programare de validare care să realizeze cel puțin verificările logice asupra datelor pe care le subînțelege un control uman (un om nu va admite că un strung produce 5,3 șuruburi, însă un program greșit întocmit va considera normală valoarea respectivă). În plus, sistemul automat de culegere va putea realiza verificări superioare unui operator uman, atît calitativ cît și cantitativ. Validarea va fi realizată și ea în timp real, adică în același timp cu culegerea datelor.

În al treilea rînd, sistemele automate de culegere oferă posibilitatea ca datele să fie prelucrate și pregătite pentru a fi ulterior utilizate direct în alte aplicații sau într-o bază de date (sortări, completări de cîmpuri, modificări ale formatelor înregistrărilor etc.).

9.7.1. Sistem de culegere a datelor pentru pontaj

Terminalul de pontaj

Realizarea aplicației a fost determinată de apariția terminalelor specializate pentru pontaj, prin intermediul cărora se culeg datele necesare completării unor rubrici din fișa de pontaj (fig. 1).

Datele personale ale fiecărui om al muncii (numărul de marcă și codul întreprinderii sau secției) sînt înscrise codificat pe o legitimație din plastic de dimensiunile 98,5×50×2,5 mm.

În afara acestor date care pot fi citite optic de către terminalul de pontaj, pe legitimație mai sînt înscrise numele și prenumele posesorului și numele întreprinderii. De asemenea, pe legitimație mai este prevăzut și un loc pentru fotografie (2×3 cm).

Terminalul de pontaj este cuplat pe o linie telegrafică de maximum 2 km lungime. Pe o linie se poate cupla doar un singur terminal, configurația rezultată pentru întreg sistemul fiind de tip stea.

Mesajele transmise microcalculatorului sau primite de la acesta sînt codificate ASCII — 7 biți.

Terminalul de pontaj citește datele codificate pe legitimație prin intermediul unui bloc optic. După citire, cifrele de marcă sînt introduse într-un registru de memorie și transmise prin linia telegrafică microcalculatorului. Terminalul nu ia în considerare decît legitimațiile care au codul corespunzător întreprinderii (secției) respective. De asemenea, datele sînt

preluate numai dacă blocul de citire sesizează (optic) introducerea corectă a legitimației. Blocul optic nu mai citește datele de pe această legitimație, decît după extragerea ei completă din terminal.

Numărul de marcă citit de pe legitimație este codificat de terminal într-un mesaj de lungime fixă, format din caractere ASCII. Fiecare caracter al mesajului corespunde unei cifre din numărul de marcă și este format dintr-o parte reprezentînd tipul mesajului (codificat pe trei din biții cei mai semnificativi) și o parte de dată (cifrele 0—9 codificate binar pe ultimii patru biți).

În unele tipuri de mesaje partea de dată nu este luată în considerare (biții componenți vor fi notați în cele ce urmează cu x).

Mesajele transmise sau primite de la microcalculator au următoarea codificare :

— către microcalculator :
  — caracter de cifră de marcă citită, terminalul fiind în starea „INTRARE" ;
  — caracter de cifră de marcă citită, terminalul fiind în starea „IEȘIRE" ;
  — mesaj de eroare ;
  — terminal conectat (pus sub tensiune) ;
  — terminal deconectat ;

— de la microcalculator :
  — caracter de marcă (ecou) ;
  — mesaj care conține cifrele de timp ;
  — mesaj de trecere pe starea „INTRARE" ;
  — mesaj de trecere pe starea „IEȘIRE" ;
  — mesaj de stingere.

După transmiterea numărului de marcă terminalul așteaptă confirmarea de la microcalculator (ecou).

La primirea mesajului de confirmare se compară partea de dată din fiecare caracter cu partea de dată păstrată în registrul de memorie al terminalului.

În cazul coincidenței, terminalul emite un semnal sonor (1 kHz) cu durata de o secundă, prin intermediul unui difuzor miniatural încorporat. În același timp pe ecranul terminalului apar cifrele de marcă (confirmarea optică a preluării corecte a datelor de către microcalculator), înlocuind pentru o secundă cifrele de timp afișate.

Dacă numărul de marcă primit de la microcalculator diferă de cel existent în registrul terminalului, datorită cel mai probabil apariției unor erori de transmisie pe linie, atunci terminalul nu mai emite semnalele de confirmare (optim și sonor) și avertizează microcalculatorul, printr-un mesaj de eroare, că numărul de marcă primit anterior nu trebuie luat în considerare (fig. 9.47).

Ecranul terminalului de pontaj este format dintr-un șir (baghetă) de LED-uri numerice, care afișează în permanență cifrele de timp (ora și minutul) și o zonă unde apare luminat fie cuvîntul „INTRARE" fie „IEȘIRE", corespunzător tipului de pontaj care poate fi executat la terminalul respectiv.

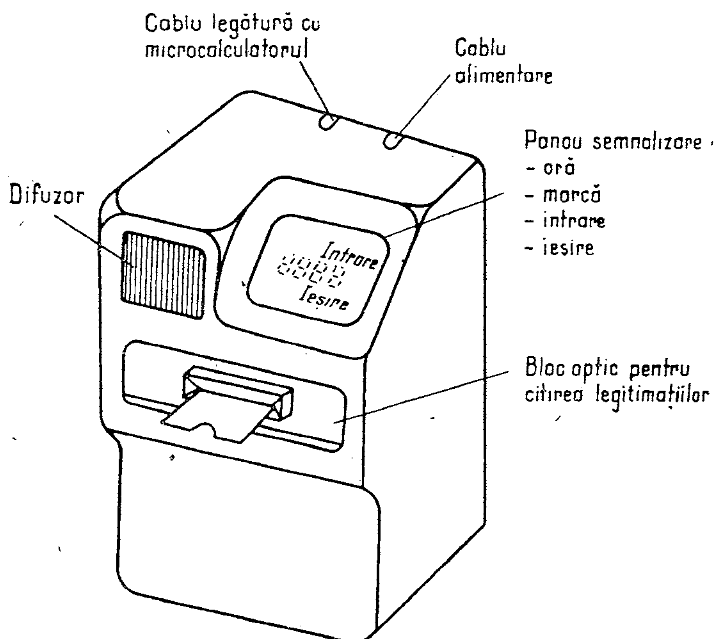


Fig. 9.47. Terminalul de pontaj.

Terminalul de pontaj nu are ceas încorporat, fiind prin urmare mai simplu şi deci mai ieftin. De aceea el trebuie să primească cifrele de timp de la microcalculator şi să le afişeze în permanenţă pe ecran.

Terminalul de pontaj emite la punerea sub tensiune şi respectiv la scoaterea sa de sub tensiune cîte un mesaj.

De asemenea terminalul recunoaşte un mesaj de stingere de la calculator (se acţionează un releu încorporat care întrerupe legătura de reţeaua de alimentare).

Monitor pentru gestiunea terminalelor de pontaj

Programul monitor pentru gestiunea terminalelor de pontaj se execută pe microcalculatorul Felix M-18 sub controlul sistemului de operare RTOS 80.

Monitorul realizează următoarele funcţii :

— gestionează activitatea terminalelor de pontaj cuplate la microcalculator şi comunică operatorului date referitoare la funcţionarea sistemului ;

— poziţionează terminalele din configuraţie, la comanda operatorului, pe starea INTRARE sau IEŞIRE ;

— comunică timpul curent terminalelor şi-l actualizează minut cu minut ;

— primeşte cifrele de marcă, face verificarea lor (trebuie să fie numerice) şi transmite mesajul de confirmare terminalului de pontaj ;

— formează înregistrări cu mărcile primite, ataşîndu-le şi alte informaţii necesare pontajului şi înscrie înregistrările sub forma unui fişier tip SFDX pe disc flexibil.

Monitorul este format dintr-un program principal şi mai multe rutine corespunzătoare funcţiilor care trebuie îndeplinite în gestionarea activităţii terminalelor. Se menţionează că toate cererile de intrare/ieşire (QIO) sînt înaintate executivului RTOS.

Acesta formează o coadă de aşteptare de tipul primul venit primul servit (FIFO). Monitorul de gestiune a terminalelor de pontaj trebuie realizat în aşa fel încît să funcţioneze corect în condiţiile menţionate. În continuare se vor prezenta cîteva din caracteristicile monitorului.

Interfaţa cu operatorul sistemului. Monitorul este încărcat şi lansat de către operator, după ce mai întîi a declarat terminalele de pontaj ca fiind terminale „slave" (adică nu transmit mesaje terminate prin caracterul <CR>), iar data (an, luna, zi, oră şi minut) a fost comunicată sistemului.

În continuare are loc un dialog cu operatorul, în scopul stabilirii condiţiilor de lucru şi a configuraţiei sistemului.

În această etapă operatorul stabileşte numărul terminalelor de pontaj din configuraţie şi starea lor. Monitorul verifică rezolvarea cererilor de intrare/ieşire şi stinge terminalele care nu răspund corect comenzilor transmise, comunicînd în acelaşi timp şi operatorului măsurile luate. Apoi monitorul verifică existenţa discului flexibil pe care urmează să fie stocate datele referitoare la pontaj ; în caz afirmativ deschide un fişier a cărui primă înregistrare conţine data creării fişierului (an, luna, zi, ora, minutul, secunda).

La cererea operatorului, noile înregistrări pot fi ataşate în continuarea unui fişier cu mărci existente deja pe discul flexibil. Evident că în acest caz nu mai are loc deschiderea unui alt fişier de mărci şi nici înscrierea datei.

Dacă nu au apărut erori, monitorul anunţă operatorul printr-un mesaj la consolă despre începerea activităţii de gestionare a terminalelor.

Pe durata funcţionării monitorului, operatorul este înştiinţat despre evenimente ca : primirea unui număr de marcă în care există caractere nenumerice, anularea unui număr de marcă de către un terminal prin mesajul de eroare marcă (se depistează astfel liniile de comunicaţie cu erori), conectarea sau deconectarea unui terminal de pontaj la, respectiv de la reţea, erori la scrierea înregistrărilor pe disc sau umplerea discului flexibil etc.

În situaţia evenimentelor care privesc terminalul de pontaj monitorul afişează la comandă date referitoare la : numărul terminalului care este implicat, starea sa (intrare sau ieşire) ora şi minutul producerii evenimentului şi numărul de marcă transmis de terminal.

La terminarea unei perioade de pontaj (maximum 24 ore) operatorul comandă încheierea execuţiei monitorului. Ca urmare fişierul cu mărci este închis, terminalele afişează pe ecrane un şir de zerouri, iar operatorul primeşte mesajul de confirmare a terminării corecte a execuţiei.

Prelucrarea mesajelor care conţin numere de marcă este declanşată de sosirea unui mesaj de la unul dintre terminalele de pontaj din configuraţia sistemului. Pentru fiecare terminal instalat în sistem există în cadrul monitorului un număr de instrucţiuni cărora li se dă controlul prin mecanismul de AST (asinchronous system trap). După analizarea zonei de cod din primul caracter al mesajului, în cazul constatării că a sosit un

mesaj conținînd un mesaj de marcă se preiau și restul de caractere. Apoi are loc controlul zonelor de date din caracterele de marcă și stabilirea stării terminalului care a transmis mesajul. Dacă s-au depistat caractere nenumerice în marcă, prelucrarea este abandonată, iar monitorul, după semnalarea erorii, reface cererea de așteptare mesaje de la terminalul respectiv.

Dacă mesajul primit a fost corect, se pregătește mesajul de răspuns către terminalul respectiv și se expediază, urmînd o perioadă de așteptare de 100 ms (mecanismul de mark time) cu o cerere de intrare/ieșire (QIO) pentru eventualul mesaj de eroare.

Dacă terminalul răspunde cu mesaj de eroare (1 caracter), atunci se semnalează operatorului și monitorul reface cererea (QIO) de așteptare mesaje de la terminal. Dacă s-au scurs cele 100 ms, atunci înseamnă că numărul de marcă primit poate fi înregistrat în fișier și se dă controlul rutinei care gestionează fișierul de mărci.

Înregistrarea datelor în fișierul de mărci se face în cod ASCII. Ca urmare, înainte de formarea înregistrării este necesară o operație de conversie. Înregistrarea are cîmpurile : numărul de marcă, numărul logic al terminalului de pontaj care a citit și a transmis marca, starea terminalului în momentul citirii mărcii, ora și minutul citirii. Opțional fiecare înregistrare se termină cu caracterele <CR> <LF> necesare prelucrării în COBOL a fișierului de mărci.

Înregistrarea astfel formată este trecută în fișier. Rutina gestionează toate erorile care pot apare în timpul lucrului cu fișierul de mărci.

Transmiterea (actualizarea) timpului la terminalele de pontaj a ridicat probleme de programare care nu și-au găsit o rezolvare imediată. Astfel, s-a mai amintit despre ordonarea în coadă de așteptare a cererilor de intrare/ieșire înaintate executivului sistemului de operare.

Inițial, pentru simplitate, s-a folosit un task separat pentru actualizarea timpului. La fiecare 60 de secunde taskul intra în acțiune, forma mesajele de timp cu cifrele de oră și minut și le transmitea terminalelor. Sistemul funcționa corect atîta timp cît se ponta la cel puțin o dată pe minut la fiecare terminal instalat în sistem. Dacă nu se efectua pontajul minute în șir terminalele nu schimbau timpul la fiecare minut. După un simplu pontaj însă, terminalele arătau ora exactă ! Explicația este următoarea : cererile de citire a mărcilor de la terminale blochează terminalele de pontaj și nu mai permit accesul mesajelor de timp lansate la fiecare minut, care astfel ajung să se acumuleze într-o coadă de așteptare (chiar dacă taskul care le lansează are o prioritate mai mare). Este suficient însă un singur pontaj, pentru ca terminalul să se elibereze și să permită mesajelor conținînd timpul să ajungă la el și să afișeze rapid toate cifrele cu ora și minutul, pe ecranul terminalului rămînînd în final numai timpul din ultimul mesaj aflat în coadă, adică tocmai ora exactă !

Problema a fost rezolvată prin introducerea funcției de actualizare a timpului la terminal în interiorul monitorului și efectuarea următoarelor operații, la fiecare 60 de secunde, în cadrul unei rutine specializate :

a) pregătirea cifrelor de timp corespunzătoare codului recunoscut de terminalul de pontaj și, în același timp, a cifrelor codificate ASCII pentru a figura în înregistrările din fișierul de mărci ;

b) anularea tuturor cererilor de intrare/ieșire lansate terminalelor de pontaj, inclusiv a celor aflate în curs de execuție ;

c) lansarea mesajelor de actualizare a timpului de pe ecranele terminalelor ;

d) refacerea cererilor de intrare pentru mărcile de la terminalele de pontaj ;

Evident că în momentul activării rutinei de actualizare a timpului, operațiile de la punctul b perturbă funcționarea corectă a procedurilor de lucru cu terminalele de pontaj. De fapt, sînt afectate numai cererile de intrare/ieșire în execuție, ceea ce implică în final, absența mesajelor de confirmare (sonor și optic) de la unele terminale de pontaj.

Cererile de intrare/ieșire către consola sistemului sau unitatea de disc flexibil nu sînt afectate. Prin urmare procedura adoptată cere ca personalul care pontează, introduce legitimațiile corect dar nu primește confirmarea, să mai repete o dată operația, mai ales cînd operația de introducere a legitimației a fost însoțită instantaneu de modificarea cifrelor de timp de pe ecranul terminalului.

Anularea unor cereri de intrare/ieșire lansate mai are loc atunci cînd de la terminal se așteaptă un mesaj conținînd un număr de marcă (format din mai multe caractere ASCII) și în locul lui vine un mesaj de eroare marcă sau activare terminal (format dintr-un singur caracter ASCII).

De aceea s-a adoptat metoda de a lansa două cereri consecutive de mesaj de la terminal, prima pentru un caracter și a doua pentru restul de caractere, fără timpi de așteptare între ele.

Dacă primul caracter reprezintă un mesaj care face parte din categoria mesajelor formate dintr-un singur caracter, atunci are loc anularea celei de a doua cereri de intrare/ieșire.

Existența a două cereri de intrare pentru un singur mesaj a generat o altă problemă datorită probabilității foarte mici, dar nu de neglijat, ca activarea rutinei de actualizare a timpului să aibă loc mai înainte ca cea de a doua cerere să fie activată, prima cerere fiind satisfăcută de preluarea primului caracter dintre cele n caractere care constituie numărul de marcă. În acest caz rar, dar posibil, după transmiterea timpului la terminale se refac cele două cereri de prelucrare date de la terminalul de pontaj care a apucat însă să transmită deja un caracter, pierdut datorită operațiilor de la punctul b.

Prima cerere lansată preia al doilea din cele n caractere de marcă (pe care-l consideră a fi primul), iar a doua cerere, care așteaptă să primească celelalte n—1 caractere, nu primește decît restul de n—2, rămînînd să aștepte la infinit ultimul caracter și blocînd astfel terminalul de pontaj. Pentru a preîntîmpina situațiile de acest gen, s-a impus ca pe durata execuției celor două cereri consecutive să nu fie posibilă nici o întrerupere (disabile interrupt).

Au fost prezentate doar cîteva din problemele ridicate de realizarea monitorului pentru gestiunea terminalelor de pontaj. În ciuda operațiilor prezentate, aparent complicate și foarte consumatoare de timp, experimentarea monitorului într-un sistem cu 16 terminale de pontaj, la care 16 persoane au introdus legitimații, într-un ritm cu mult superior aceluia realizat de o coadă de oameni care trec prin fața terminalului și pontează, a arătat funcționarea ireproșabilă a sistemului de pontaj. Performanța sis-

temului nu este practic afectată de numărul mai mare sau mai mic de terminale de pontaj existente în configurație.

Un fișier de mărci, creat pe un disc flexibil gol, poate ajunge la cca 10 000 înregistrări de pontaj.

9.7.2. Sistem de culegere a datelor cu terminale industriale specializate

Terminalul industrial specializat pentru culegere

Terminalul poartă denumirea „industrial" deoarece este destinat în primul rînd culegerii de date din întreprinderi ; se numește specializat deoarece, prin modulele care-l compun și funcțiile pe care le realizează, este în mod particular potrivit activității de culegere a datelor.

În continuare se vor descrie modulele din configurația terminalului industrial specializat, denumit pe parcursul paragrafului doar „terminal" (fig. 9.48).

Tastatura numerică este formată din tastele pentru cifrele 0—9 punctul zecimal și semnul minus. Aceleași taste, în combinație cu alte trei taste (CTRL A-I, CTRL J-O, CTRL P-Z) generează codurile ASCII ale literelor. De aceea, fiecare tastă numerică are înscris pe ea și trei litere ale alfabetului.

Tastatura funcțională este formată din următoarele taste :

— a, b, c, d, e, f — numite și taste de tranzacții, prin care utilizatorul selecționează tipul tranzacției, adică programul din microcalculator sub controlul căruia se vor desfășura operațiile de culegere a datelor de la terminalul respectiv.

— ENTER — care semnalează sfîrșitul introducerii datelor pentru un cîmp al tranzacției.

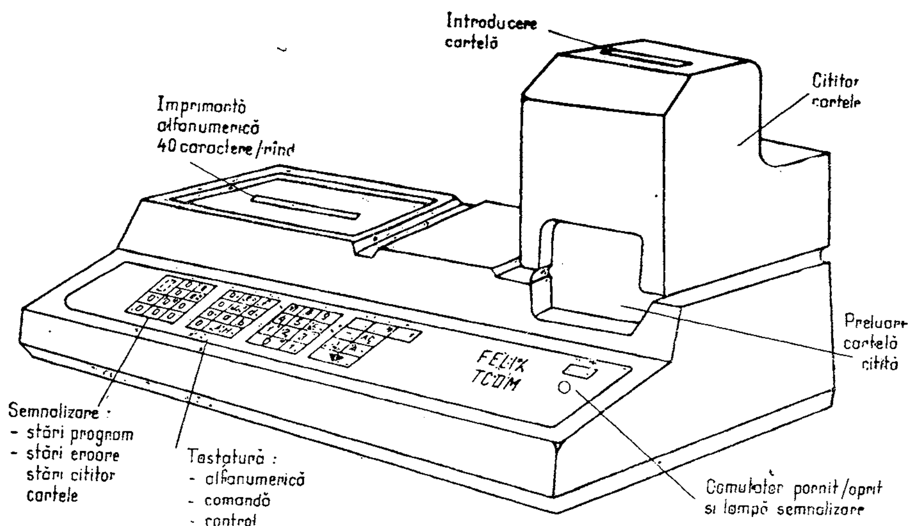


Fig. 9.48. Modulele din configurația terminalului industrial specializat.

— END — care apăsată semnalează monitorului de culegere a datelor că s-a terminat o „sesiune de lucru“ (formată din rularea, cel puțin o dată, a unuia din programele de control a culegerii datelor, selectat printr-o tastă de tranzacție în conformitate cu aplicația dorită).

— CC și CA — au rolul de a anula ultimul caracter tastat, respectiv ultimul cîmp introdus.

— INIT — semnalează monitorului actualizarea terminalului, intenția operatorului de a începe o sesiune de lucru.

— CLEAR ERROR — comunică programului de culegere a datelor că operatorul a înregistrat eroarea semnalată și, în continuare, va acționa pentru înlăturarea cauzei care a provocat apariția erorii.

— PAPER FEED — avansează hîrtia de la imprimanta terminalului cu un rînd.

Imprimanta terminalului are dimensiuni reduse și ca urmare a fost încorporată în terminal. Imprimanta are capacitatea de a scrie 40 de caractere alfanumerice pe un rînd. În cadrul terminalului imprimanta are funcții multiple.

Astfel, pe imprimantă sînt înscrise, în ecou, datele introduse de utilizator de la terminal, pentru ca prin control vizual să fie depistate și eventualele erori de tastare. De asemenea pe imprimantă apar mesajele transmise operatorului de către programul de culegere care-l asistă în activitatea sa. Programul de control al culegerii de date, realizat pentru a ghida operatorul în activitatea sa la terminal, afișează mesaje prin care se solicită introducerea cîmpurilor tranzacției, în ordinea stabilită.

Totalitatea mesajelor transmise de la microcalculator, împreună cu datele tastate sau introduse de la terminal prin intermediul cartelei perforate, rămîn înscrise pe imprimantă și constituie protocolul activității de culegere a datelor de la terminalul respectiv.

Cititorul de cartele, încorporat și el în terminal, permite preluarea datelor de pe cartele perforate în cod Hollerith.

Datele sînt citite în timpul căderii libere a cartelei introdusă cu mîna în cititor. Cititorul nu sesizează introducerea corectă a cartelei (colțul tăiat al cartelei) și de aceea a fost necesară stabilirea unui caracter, care perforat în prima coloană a cartelei, permite programului de culegere să valideze sau nu datele citite și transmise de terminal. Dacă primul caracter primit, într-un mesaj care conține date citite de pe cartelă, nu este codul ASCII al literei „O“, atunci cartela nu a fost introdusă corect și se cere repetarea operației.

Ghidul utilizator reprezintă o unitate funcțională formată din imprimantă și un set de becuri, prin care persoana de la terminal este asistată în desfășurarea operațiilor de culegere a datelor.

După apăsarea tastei INIT, programul monitor de culegere a datelor ia cunoștință de activarea terminalului și răspunde prin aprinderea becului READY la terminal.

Terminalul mai are și un bec prin care, din program, se semnalează că următorul cîmp al tranzacției trebuie introdus de pe cartelă. După citirea cartelei becul se stinge. La depistarea unor erori locale, cartele necunoscute sau operare incorectă a cititorului de cartele, se aprind două becuri unul notat ERROR CARD și altul FEED CARD.

Tastele de tranzacții, notate a, b, c, d, e, f, au fiecare atașat cîte un bec. După selectarea tipului de tranzacție, la terminalul respectiv rămîne aprins becul corespunzător tastei apăsate, pe toată durata unei sesiuni de lucru.

Imprimanta este utilizată pentru afișarea cîmpurilor tranzacției introduse de la terminal și a mesajelor de ghidare a operatorului.

În cazul unor operații incorecte, cînd nu se respectă ordinea sau natura cîmpurilor tranzacției, în afara mesajelor scrise la imprimanta terminalului (sau în locul lor), se poate semnala tipul erorii prin intermediul unui LED numeric existent la terminal.

Cifrele afișate pe LED sînt puse în corespondență cu tipurile de erori posibile (nerespectarea lungimii cîmpurilor, introducerea de caractere numerice în cîmpuri declarate alfabetice sau invers, introducerea de date de la tastatură într-un cîmp care așteaptă date de la cititorul de cartele etc.).

Sistem de programe pentru gestiunea terminalelor de culegere

Sistemul de programe pentru gestiunea terminalelor de culegere rulează pe microcalculatorul Felix M-18 sub controlul sistemului de operare RTOS 80.

Sistemul de programe are patru componente, distincte ca funcțiuni și mod de realizare : monitorul, biblioteca de funcții primitive, biblioteca de rutine de validare și programele de culegere.

Monitorul are sarcina gestiunii activității terminalelor de culegere a datelor instalate în sistem. În acest scop monitorul creează pentru fiecare terminal un „context“, format din mai multe liste de tip stivă, pe care le administrează. Monitorul realizează asocierea dintre terminal și programul de culegere solicitat de operator prin intermediul uneia din tastele de tranzacții, execută baleierea tuturor terminalelor și alocă timp de prelucrare din partea unității centrale pentru programele de culegere activate. Monitorul este scris în limbaj de ansamblare din considerații legate de performanță. Monitorul are două intrări distincte. *Prima intrare* este punctul de lansare a întregului program, secvența în care sînt inițializate toate tabelele (context) pentru terminalele declarate în sistem și se activează programul care preia tranzacțiile introduse și le depune într-un fișier pe bandă magnetică. Apoi monitorul intră într-o buclă infinită de baleiere a terminalelor, așteptînd o întrerupere software de tip AST, situație similară cu activitatea monitorului de gestiune a terminalelor de pontaj. Adresa de AST este unică pentru fiecare terminal.

La primirea unei întreruperi software de la terminal monitorul caută în tabela de stare a terminalelor (TST) pentru a vedea dacă terminalul respectiv a fost inițializat (adică a fost marcat ca fiind în stare de lucru) sau nu.

Dacă a fost inițializat, atunci înseamnă că întreruperea semnalează terminarea unei operații de intrare/ieșire. Ca urmare în tabela terminalului respectiv se marchează terminarea operației I/E, se reface adresa programului de culegere care așteaptă datele și apoi se relansează execuția sa.

Dacă terminalul nu a fost inițializat, atunci se lansează rutina de inițializare (care stabilește, în dialog cu operatorul terminalului, care este programul de culegere solicitat) și apoi se marchează în tabela TST că terminalul a fost inițializat.

Deci, după ce monitorul servește întreruperile software, și operează în conformitate cu datele din tabela TST, dă controlul unui program (de culegere sau inițializare) și își încetează activitatea, adică rămîne în așteptare (buclă infinită de baleiere a terminalelor).

Un program de culegere a datelor are afectat timpul unității centrale, în exclusivitate, pînă în momentul în care se lansează o cerere de intrare/ieșire. Apoi se poziționează bitul de așteptare a unei întreruperi AST, în tabela TST și se dă controlul monitorului care, în continuare, parcurge toate terminalele, dînd sau nu controlul programelor atașate acestora, în funcție de starea programelor.

Revenirea la primul terminal nu se face decît după ce au fost interogate și eventual relansate toate programele de culegere aflate în legătură cu celelalte terminale ale sistemului.

Între terminalele de culegere nu există nici un fel de prioritate, toate terminalele sînt egale pentru monitor. În plus, atîta timp cît sînt active programele asociate lor, sînt stăpîne pe întreg sistemul în ansamblu. Totuși nu se justifică introducerea unui mecanism de partajarea timpului (time sharing), chiar dacă există pericolul ca un program de culegere a datelor, întocmit greșit, să bucleze într-o operație de intrare/ieșire și să blocheze orice altă activitate a sistemului. Experimental s-a constatat că sistemul se comportă bine, cu timpii de așteptare la terminal sub 2 secunde, deoarece în mod normal, un program de culegere a datelor oricît de complicat ar fi, nu poate avea timpul mediu între două operații de intrare/ieșire, de natură să afecteze timpii de așteptare la terminal.

A doua intrare în monitor este secvența de program care pune în așteptare programul de culegere atașat unui terminal în lucru; acest lucru se întîmplă în momentul în care se execută o operație de intrare/ieșire. În stiva (context) a terminalului, notată CTX, se salvează adresa din programul de culegere la care trebuie să se revină după terminarea operației de intrare/ieșire solicitată de programul de culegere respectiv.

După aceea monitorul continuă baleierea terminalelor în modul descris mai înainte pentru prima intrare.

Tabela CTX conțin 256 octeți și există cîte o tabelă pentru fiecare terminal instalat în sistem. În fiecare tabelă există înscris numărul logic atașat terminalului, cu adresa AST atașată terminalului, pointerul curent și pointerul de început al stivei. Practic, în tabela CTX se acumulează toate datele (cîmpurile) care compun o tranzacție și se stochează temporar informații referitoare la programul de culegere (salvări de adrese și variabile de lucru, deoarece programele de culegere fiind reentrante nu au o zonă de date fixată în memorie).

Monitorul gestionează umplerea și golirea stivelor; el permite lucrul cu tranzacții care nu depășesc 249 de octeți.

Biblioteca de funcții primitive are rolul de a ușura munca programatorului deoarece rutinele conținute se referă la o serie de comenzi și operații care se repetă des în cadrul programelor de culegere a datelor. Rutinele sînt secvențe de cod realizate ca module reentrante.

Reentranța modulelor este absolut necesară pentru a permite lucrul a mai multor terminale, sub controlul aceluiași program de culegere și pentru a evita existența mai multor copii în memoria internă a microcalculatorului.

* Rutina de inițializare a dialogului cu terminalul este apelată atunci cînd monitorul a primit de la un terminal codul corespunzător apăsării tastei INIT. Rutina comandă aprinderea becurilor a, b, c, d, e, f, corespunzătoare tastelor de tranzacții ale terminalului și așteaptă apăsarea unei taste de tranzacție (a—f). După aceea rămîne aprins doar becul tastei selectate și se lansează în execuție programul de culegere corespunzător. Dacă în loc de o tastă (a—f) s-a apăsat din nou tasta INIT, atunci se sting toate becurile și inițializarea este reluată cu aprinderea din nou a becurilor de tranzacții.

* Rutina de citire a unei cartele de la terminal are ca parametrii adresa de memorie la care se vor depune caracterele preluate și un indicator care arată dacă datele trebuie transmise sau nu în ecou la imprimanta terminalului.

* Rutina de preluare a unui cîmp (tastat) de la terminal cere specificarea : adresei de depunere a caracterelor preluate, a lungimii maxime a cîmpurilor și dacă este necesară scrierea cîmpului la imprimanta terminalului (ecou).

* Rutina de scriere a unui mesaj la terminal cere precizarea adresei unde se află mesajul și lungimea mesajului.

* Rutina de aprindere și cea de stingere a unui bec de la terminal are ca unic parametru codul becului.

* Rutina de aprindere a becului de eroare și înscrierea unei cifre de eroare corespunzătoare pe LED-ul numeric are ca parametru cifra care trebuie afișată pe LED.

Mai sînt și alte rutine în biblioteca de funcții primitive, toate fiind prevăzute cu un cod de retur în programul apelant, cod care specifică îndeplinirea cu succes a funcției pentru care rutina a fost solicitată.

Biblioteca de rutine de validare conține mai multe module care realizează controlul datelor din cîmpurile tranzacției. La apelarea rutinelor, programul de culegere trebuie să specifice ca parametrii, adresa și lungimea cîmpurilor care trebuie verificat.

Codul de retur este o variabilă care conține fie valoarea „adevărat" fie valoarea „fals", în funcție de rezultatul verificării.

Validarea datelor din cîmpurile tranzacției se face în timp real, adică imediat după depunerea datelor la adresa specificată. În cazul invalidării datelor se semnalează eroarea la terminal și datele pot fi corectate imediat de către operatorul terminalului.

Rutinele de validare sînt constituite într-o bibliotecă rezidentă pe discul flexibil. Utilizatorul își poate dezvolta rutinele de validare proprii, conform tranzacțiilor particulare cu care lucrează sistemul de culegere.

Biblioteca de rutine de validare conține un repertoriu minim, capabil să valideze cîmpurile numerice (ca natură, ca încadrare între două limite specificate ca parametrii, ca existență într-o listă finită de valori), cîmpuri alfabetice (ca natură) și cîmpuri alfanumerice (compararea a două șiruri de caractere, compararea unui octet cu toate caracterele unui șir de caractere).

Programele de culegere. Fiecare utilizator își dezvoltă programele proprii de culegere, în limbajul PL/M, cu atributul REENTRANT și PUBLIC, pe baza funcțiilor primitive declarate cu EXTERNAL și a rutinelor de validare.

Sistemul de culegere a datelor cu terminale industriale specializate este realizat în mod „deschis“, utilizatorul poate să-şi scrie propriile rutine de validare sau funcţii primitive ; el poate renunţa la unele din modulele de bibliotecă prezentate mai înainte, care nu-i sînt necesare, în funcţie de structura tranzacţiilor proprii, micşorînd la maximum zona de memorie ocupată de programe.

Tranzacţiile validate sînt depuse într-un fişier pe bandă magnetică creat conform standardului ANSI.

În continuare se va prezenta o schiţă de tranzacţie, simplă, numită „tranzacţie tip a“, care are două cîmpuri, unul introdus de pe cartelă, „nume articol“ şi altul de la tastatura „cantitatea“ (4 caractere numerice). Se cere verificarea existenţei caracterelor numerice în cîmpul „cantitate“.

Programul de culegere trebuie să parcurgă următorii paşi :

1. Declararea variabilelor. Descrierea mesajelor.
2. Transmiterea mesajului „tranzacţie tip a“ şi afişarea sa pe imprimanta terminalului.
3. Pregătirea preluării numelui articolului de pe cartelă, şi aprinde becul „CARD REQUEST“ la terminal.
4. Citirea datelor de pe cartelă. Verificarea introducerii corecte a cartelei. Dacă da, atunci se stinge becul aprins în pasul anterior. Dacă nu, se trece la pasul 3 şi pe LED-ul numeric apare cifra 7 semnalizînd tipul erorii.
5. Transmiterea mesajului „cantitate“ la terminal şi scrierea la imprimantă.
6. Preluarea a maximum patru caractere numerice (unul, două sau trei caractere urmate de ENTER sau patru caractere fără a mai fi necesară apăsarea tastei ENTER).
   Afişarea lor la imprimanta terminalului.
7. Validarea cîmpului primit, toate caracterele trebuie să fie numerice. Dacă se îndeplineşte condiţia de validare tranzacţia se termină ; dacă nu se merge la pasul 8.
8. Mesaj la terminal pe imprimantă „cîmp numeric“, iar pe LED-ul de eroare trebuie să apară înscrisă cifra 5. Salt la pasul 6.

Scrierea programului de culegere de mai sus solicită un efort minim din partea programatorului. În paşii 2, 5 şi 8 se apelează rutina de transmisie a unui mesaj la terminal ; în paşii 3, 4 şi 8 se apelează rutinele de aprindere şi stingere a unor becuri ale terminalului. Paşii 4 şi 6 necesită apelarea rutinelor de preluare a datelor de pe cartelă, respectiv de la tastatura terminalului. Validarea numerică este solicitată în pasul 7.

Tranzacţii mai complicate nu conduc la o complexitate ridicată a programelor de culegere deoarece pentru fiecare cîmp al tranzacţiei sînt parcurse în principiu patru faze : pregătirea cîmpului, preluarea datelor, aplicarea condiţiilor de validare şi comunicarea erorilor (dacă au apărut).

Prin urmare programul de culegere reprezintă o structură liniară de secvenţe de cod, fiecare secvenţă urmărind preluarea datelor corespunzătoare unui cîmp.

Rezolvarea condiţiilor de validare pentru cîmpuri ale căror date sînt corelate rămîne în sarcina programatorului, aceste tipuri de tranzacţie avînd o frecvenţă redusă în aplicaţiile practice de culegere a datelor.

9.7.3. Configuraţia sistemului de culegere a datelor

Sistemul de echipamente este alcătuit din :

— microcalculatorul Felix M 18/M 18 B ;

— multiplexor MX cu căi telegrafice pentru terminalele de culegere ;

— consolă sistem ;

— unitate de bandă magnetică ;

— unitate de disc flexibil ;

— 1—16 terminale de culegere (pontaj sau industriale specializate) ;

— linii telegrafice de comunicaţii (3 fire).

Figura 9.49 prezintă configuraţia de sistem de culegere date (software şi hardware) şi pontaj automat.

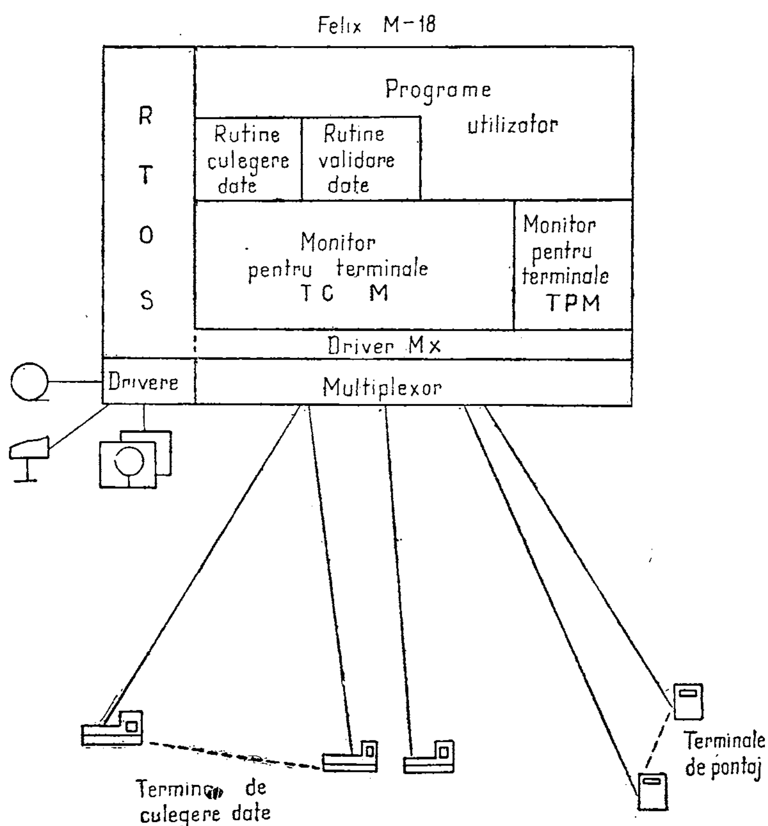


Fig. 9.49. Configuraţia de sistem de culegere date.

9.8. Sistemul S.C.V.D. de introducere şi validare a datelor cu ajutorul microcalculatorului Felix M-18

9.8.1. Prezentare generală

Sistemul a fost elaborat în scopul de a pune la dispoziţia utilizatorilor un instrument de lucru eficient în colectarea, validarea, stocarea datelor pe suport magnetic.

Utilizatorii dispun în plus de un set de funcţii pentru gestionarea fişierelor conform standardului ANSI.

Introducerea datelor în sistem se poate face de la mai multe terminale, de la consola sau cititorul de cartele a sistemului FELIX M-18.

Sistemul de introducere şi validare a datelor este de tip interactiv, avînd încorporate în el atît componente conversaţionale, cît şi tranzacţionale.

Astfel de componente conversaţionale sînt :

— funcţia „LISTARE“ a fişierului de ieşire ;

— funcţia de „ACCES“ la fişierul de ieşire ;

iar de componente tranzacţionale :

— introducerea interactivă şi validarea datelor ;

— funcţia de „MODIFICARE“ la nivel de caracter a articolelor din fişierul de ieşire.

Articolele (tranzacţiile) supuse validării sistemului au ca diviziune de bază „cîmpul“. Fiecare cîmp este supus validării independent de poziţia sau conţinutul altor cîmpuri, cît şi în relaţie cu celelalte cîmpuri din articol.

Pentru realizarea validării cîmpurilor, utilizatorul va scrie un program într-un limbaj specializat.

Acest program este compus din instrucţiuni foarte simple ; cu unele excepţii, fiecărui cîmp i se asociază o instrucţiune.

Un astfel de program de validare împreună cu un cap de tabel afişabil pe terminal sau imprimantă formează ceea ce numim un fişier format.

Mulţimea tuturor acestor fişiere reprezintă colecţia fişierelor de intrare şi este organizată aprioric introducerii datelor pe un suport magnetic din sistemul de calcul.

În concluzie, validarea unui set de tranzacţii se realizează sub controlul unui fişier format apelat din colecţie.

Problemele legate de organizarea şi întreţinerea colecţiei de fişiere format se rezolvă de un utilizator privilegiat, iar accesul la consultarea fişierelor din colecţie este permis oricărui operator.

În cazul introducerii de la terminale a articolelor utilizatorul dispune de un set de taste programate, reprezentînd facilităţi de editare.

Articolele introduse şi validate vor fi stocate într-un fişier de ieşire organizat pe bandă magnetică, compatibil cu sistemele de calcul FELIX C-256 sau INDEPENDENT-100.

În cazul multiutilizator fiecărui articol i se asociază un identificator de utilizator, iar în final printr-un proces de partajare se va obţine pentru fiecare utilizator un fişier cu propriile articole introduse.

Fișierul de ieșire poate fi prelucrat independent de procesul de introducere/validare prin funcții de „SALVARE“ în caz de incident hardware fatal, „LISTARE“, „MODIFICARE“ și „SORTARE“.

Limbajul de validare utilizat în sistemul SCVD poate fi extins în afara instrucțiunilor standard prin instrucțiuni propuse și implementate de utilizator în conformitate cu propriile cerințe de validare.

Sistemul SCVD este implementat pe microcalculatorul FELIX M-18 și acceptă ca periferice de intrare terminale tip mașină de scris (CENTRONICS), tip display (DAF 1001/1002 S/2010) și cititorul de cartele.

Articolele găsite corecte în procesul de validare sînt listate la imprimantă în ordinea completării blocurilor din fișierul de ieșire și numerotate în ordinea scrierii lor în fișier.

Consultarea acestei liste este utilă în execuția funcțiilor „MODIFICARE“ și „SALVARE“.

Cîmpurile unui articol detectate de sistemul SCVD ca fiind eronate sînt subliniate cu caractere distincte la introducerea de la terminal, iar în cazul utilizării ca periferic de intrare a cititorului de cartele, sublinierea se face sub articolul tipărit la imprimantă.

În cazul în care configurația sistemului de calcul cuprinde și multiplexorul, numărul de terminale gestionate de SCVD este determinat de capacitatea memoriei sistemului de calcul și de sistemul de operare adoptat.

Operatorii terminalelor de tip display pot utiliza setul de taste programate care permit efectuarea corecțiilor cîmpurilor eronate fără a relua în întregime introducerea articolului.

Operațiile asupra colecției de fișiere format se execută interactiv de la consola sistemului, iar consultarea colecției se poate face și de la terminal.

Dacă se utilizează terminale de tip display, în urma consultării colecției de fișiere format, partea afișabilă din fișierul format este păstrată permanent pe ecran în regim protejat, videoinvers.

SCVD necesită următoarea configurație minimă :

— 32 KB memorie RAM ;
— consola sistem ;
— imprimanta ;
— banda magnetică ;
— sistem operare : MON 18 Vl. 4 ;

la care suplimentar se poate adăuga :

— multiplexor MX 80.20 și terminale ;
— cititor de cartele ;
— casetă magnetică ;
— discul flexibil ;
— sistem operare SFDX 18 V 3.4.

9.8.2. Funcțiile sistemului SCVD

Sistemul execută un set de funcții care abordează următoarele probleme :

a) gestionarea colecției de fișiere format ;

b) organizarea fișierului de ieșire ;
c) gestionarea terminalelor ;
d) validarea articolelor și extensia setului de validări standard ;
e) funcții de editare articole.

Aceste funcții pot fi clasificate după următoarele criterii :

1) după modul de acces la lansarea funcției :
   — *funcții protejate* — execuția lor este condiționată de cunoașterea unui cuvînt cheie ;
   — *funcții neprotejate* — nu solicită cunoașterea cuvîntului cheie.
2) după modul de lansare a funcției :
   — *funcții de sine stătătoare* — lansarea în execuția independentă de lansarea altor funcții ;
   — *funcții condiționate* — lansarea în execuție este determinată de sfîrșitul de execuție al altor funcții.

Cuvîntul cheie reprezintă o combinație de trei caractere ce sînt introduse la cererea sistemului. În cazul funcțiilor protejate sistemul recunoaște o singură cheie, stabilită la generarea sistemului.

Apelarea funcțiilor de sine stătătoare se realizează prin selectarea codului funcției din lista de funcții oferite de sistem.

Pentru execuție sînt cerute interactiv și alte informații.

Gestionarea colecției de fișiere format

Această colecție este organizată pe casetă, bandă magnetică sau disc flexibil.

Controlul asupra fișierelor din colecție se realizează printr-un fișier director, care în cazul organizării colecției pe disc flexibil este chiar fișierul director al discului. Pentru celelalte două periferice casetă și bandă magnetică prin funcția „ORGANIZARE INITIALĂ" se creează un fișier care îndeplinește rolul fișierului director. Acest fișier va fi automat poziționat ultimul în cadrul organizării secvențiale și își păstrează această poziție indiferent de fișierele incluse sau șterse ulterior organizării inițiale.

Informațiile conținute în fișierul director al colecției organizate pe bandă sau casetă magnetică sînt :

— numele fișierelor format din 3 caractere ;
— un byte pentru marcare fișier valid ;
— un byte pentru poziția fișierului în colecție.

Introducerea unui fișier în colecție se realizează prin funcția „CREARE FISIER FORMAT" care va actualiza fișierul director.

În crearea unui fișier se disting două etape :

— introducerea formatului afișabil alcătuit dintr-o matrice de 8×80 caractere ;
— introducerea programului de validare.

Sistemul SCVD asigură pentru fiecare instrucțiune introdusă, analiza sintactică și semantică. În cadrul unui apel se poate realiza o creare multiplă de astfel de fișiere format.

Prin funcția „CONSULTARE FISIER FORMAT" se selectează fișierul cerut prin nume fișier format, se afișează formatul și se memorează programul de validare. În cazul cînd nu există un fișier corespunzător numelui sau fișierul este invalidat este atenționat operatorul.

Ştergerea din colecţie a unui fişier se realizează prin funcţia „INVALIDARE FISIER FORMAT“ şi constă în modificarea byte-lui de validare cînd organizarea colecţiei este secvenţială sau în ştergerea din fişierul director al discului flexibil a informaţiilor despre fişier. Colecţia de fişiere poate conţine maximum 200 de fişiere.

În cazul organizării secvenţiale a colecţiei de fişiere format, timpul de căutare este mărit inutil de prezenţa unor fişiere invalidate logic anterior, de aceea este recomandabilă în astfel de cazuri o reorganizare la nivel fizic.

Gestionarea fişierului de ieşire

Problemele ce sînt rezolvate prin funcţiile fişierului de ieşire sînt :

— crearea pe bandă magnetică a unui fişier în format ANSI ;

— accesul la un fişier deja creat şi poziţionarea pe sfîrşitul fişierului spre a se adăuga noi articole ;

— salvarea fişierului de ieşire în caz de incidente ;

— modificări în fişierul de ieşire ;

— listarea fişierului de ieşire ;

— sortarea fişierului de ieşire.

Funcţia „CREARE FISIER IESIRE“ este de tip neprotejat şi condiţionată de primul apel al funcţiei „CONSULTARE FISIER FORMAT“.

Execuţia ei constă în crearea pe bandă magnetică a unui fişier secvenţial, în format fix. Interactiv sînt ceruţi parametrii de definire ai fişierului. Operatorul are posibilitatea de a adăuga articole la un fişier deja creat prin apelul funcţiei de „ACCES FISIER IESIRE“.

În orice situaţie în care fişierul de ieşire nu s-a putut închide, prin execuţia funcţiei „SALVARE FISIER IESIRE“ se refac etichetele de sfîrşit de fişier, folosind informaţiile conţinute în fişier, iar numărul de blocuri salvate este introdus de operator prin consultarea listei articolelor introduse.

Ca urmare a unor astfel de incidente se pierd cel mult informaţiile din ultimul bloc care se găseau în memorie şi nu se transferaseră în bandă, dar care se găsesc scrise în lista obţinută la imprimantă. Acestea se pot introduce ulterior printr-o nouă sesiune de introducere date.

Funcţia „MODIFICARE IN FISIERUL DE IESIRE“ este o funcţie protejată de sine stătătoare.

Are drept scop corectarea eventualelor erori nesesizate de SCVD în procesul de validarea datelor. Un astfel de exemplu : într-un cîmp se tastează caracterul „5“ validat prin condiţia „cuprins între limitele 4—8“, dar eronat s-a introdus caracterul „6“, această eroare nu poate fi sesizată de SCVD.

Procesul de corectare constă în :

— salvarea fişierului de ieşire începînd cu primul bloc în care se doreşte modificarea pe un fişier organizat pe o altă unitate de bandă magnetică, bandă perforată, casetă magnetică sau disc flexibil ;

— modificarea la nivel de caracter ;

— restaurarea fişierului de ieşire pe bandă magnetică.

Controlul asupra blocurilor în care se fac modificări se face prin numărul blocurilor ce apare în lista articolelor obţinută la introducere.

Secvenţa de modificare constă în introducerea de către utilizator a coordonatelor de identificare : număr bloc, număr articol din bloc, număr

caracter din articol. Utilizatorul beneficiază în execuția acestei funcții de facilitățile de editare a articolului.

Salvarea fișierului ieșire se poate executa numai dacă există spațiu suficient pentru această operație. Determinarea capacității necesare pentru realizarea salvării se face automat după prima cerere de modificare, iar în lipsă de spațiu necesar, funcția este abandonată.

Avînd în vedere principiul după care se execută operația de salvare a fișierului de ieșire și anume : de la primul bloc în care se fac modificări pînă la sfîrșitul fișierului, este recomandabil să se efectueze întîi eventuale corecții sesizate, iar ulterior să se adauge noi articole la sfîrșitul fișierului.

Ca un caz particular al apelării acestei funcții se pot realiza copii ale fișierului de ieșire pe altă bandă magnetică sau pe casetă în format M-18.

Funcția „LISTARE FISIER IESIRE“ este de tip neprotejat, de sine stătătoare.

Utilitatea funcției rezultă din necesitatea cunoașterii la un moment dat a conținutului fișierului de ieșire.

Listarea fișierului de ieșire se poate executa integral sau numai a unui număr de blocuri compacte din fișier ; de asemenea, se pot lista etichetele fișierului și se poate controla existența și poziția mărcilor de fișier. Structura listei de ieșire la imprimantă este identică cu cea obținută în urma introducerii datelor.

Funcția „SORTARE FISIER IESIRE“ este o funcție complexă de tip protejat, de sine stătătoare.

Funcția abordează următoarele probleme :

Faza A — partajarea fișierului de ieșire obținut în urma introducerii datelor în mai multe fișiere (maximum 12 fișiere la un apel).

Fiecare din aceste noi fișiere obținute este caracterizat (dacă nu este vid) prin prezența unor cîmpuri cu conținut identic la toate articolele, conținut ce a constituit criteriul de partajare. Aceste noi fișiere le vom numi în continuare „fișiere de aplicație“.

Faza B — constă în sortarea unui fișier de ieșire sau de aplicație după o combinație de cîmpuri (maximum 8 cîmpuri) în ordine crescătoare sau descrescătoare.

Faza de execuție „B“ poate fi lansată independent sau în continuarea fazei „A“.

Faza C — constă în transformarea fișierului de aplicație într-un nou fișier numit „fișier de lucrări“, exploatabil pe sistemul de calcul FELIX C-256 (fișier *1).

Execuția acestei faze se poate face independent sau în continuarea fazei „A“.

Obținerea fișierului *1, constă într-o reorganizare a articolelor pe bloc (considerate instrucțiuni și date ale unui program), avîndu-se în vedere existența cartelelor de comandă.

Gestionarea posturilor de intrare

Problema gestionării terminalelor constă pentru sistemul SCVD în a răspunde într-un timp cît mai scurt sarcinilor date de operatori. Terminalele pot fi în una din următoarele stări :

— inactiv ;
— în deschidere ;

— în lucru ;

— în închidere.

Pentru fiecare dintre stările : deschidere, lucru și închidere sînt definite tabele cu adresele liniilor, iar în funcție de apartenența unei linii la unul din tabele și poziția liniei în tabel se execută toate operațiile de comunicare pe linie cu operatorul respectiv. Terminalele sînt inițial inactive și redevin inactive după închiderea sesiunii de lucru. Trecerea din starea „inactiv“ în starea „în deschidere“ se face la inițiativa operatorului prin tastarea unui caracter prestabilit (de regulă blank).

Sistemul va trece la execuție funcției „CONSULTARE FISIER FORMAT“ pentru linia respectivă.

După ce în prealabil utilizatorul terminalului a introdus : tipul terminalului, parola utilizatorului și numele fișierului format dorit, execuția funcției „CONSULTARE FISIER FORMAT“ va duce la afișarea formatului și încărcarea programului de validare într-o zonă de memorie rezervată terminalului.

În cazul organizării secvențiale a colecției de fișiere format, deschiderea liniilor se execută secvențial, alte cereri de deschidere sînt trecute într-o coadă de așteptare, fiind activate după terminarea cererii precedente.

Facem observația că deschiderea sesiunii la un terminal nu perturbă activitatea altor terminale, cu excepția configurației cu o singură unitate de bandă (cînd colecția de fișiere format este organizată tot pe bandă magnetică), în care caz la celelalte terminale aflate în lucru se pot introduce caractere, dar acestea nu pot fi trecute în fișierul de ieșire, închis temporar, pînă nu se încheie procesul de deschidere a liniei. După încheierea execuției funcției „CONSULTARE FISIER FORMAT“, adresa liniei este ștearsă din tabela liniilor în deschidere și înscrisă în tabela liniilor în lucru.

Odată linia trecută în starea „în lucru“ operatorul poate introduce un articol, iar prin semnalarea sfîrșitului introducerii, pentru articolul și programul de validare corespunzător se execută funcția de „VALIDARE“.

La sfîrșitul unei sesiuni de lucru, operatorul comandă închiderea liniei, operație care constă în ștergerea adresei liniei din tabela liniilor în lucru și trecerea în tabela liniilor în închidere.

Odată linia închisă, adresa liniei va fi ștearsă și din tabela liniilor în închidere, linia devenind inactivă.

Operatorul poate relua procesul de a intra în comunicație cu sistemul, apelînd un alt fișier format.

Dacă toate liniile au devenit inactive sistemul SCVD interoghează operatorul de la consolă asupra continuării lucrului ; în caz afirmativ așteaptă deschiderea unei linii ; în caz contrar, fișierul de ieșire este închis, sistemul SCVD sistîndu-și execuția. Funcția „VALIDARE“ este de tip neprotejat, fiind lansată condiționat de încheierea introducerii unui articol.

Această funcție poate fi lansată secvențial de la orice terminal, eventualele cereri nesatisfăcute imediat sînt memorate într-o coadă de așteptare, fiind preluate în momentul încheierii validării unui alt articol.

În execuția funcției „VALIDARE“ fiecare cîmp este pus în relație cu instrucțiunea de validare corespunzătoare, rezultatul execuției unei instrucțiuni de validare fiind declararea unui cîmp ca fiind valid sau nu.

Articolul este declarat valid dacă toate cîmpurile sale au fost găsite valide, fiind semnalat corespunzător. Dacă există cîmpuri eronate, acestea sînt marcate ; în cazul terminalelor display cursorul este poziționat pe prima poziție a articolului, astfel încît utilizînd setul de taste programate se pot efectua corecțiile necesare, urmate de relansarea funcție de „VALIDARE“.

Introducînd articolele de la cititorul de cartele, semnalarea eventualelor erori se va face la imprimantă, iar semnalarea sfîrșitului de fișier de pe cartele se face prin .EOF.

Funcții de editare

Aceste funcții sînt neprivilegiate, de sine stătătoare. Sînt apelate prin caractere speciale și conduc la mutarea marcajului care punctează caracterul introdus în buffer-ul de editare.

Execuția acestor funcții realizează :

— deplasarea cursorului cu o poziție spre stînga ;

— afișarea pe poziția curentă a caracterului existent pe aceiași poziție în articolul anterior editat ;

— deplasarea de pe poziția cursorului a întregului articol cu o poziție spre dreapta, existînd posibilitatea de a insera un caracter pe poziția inițială a cursorului ;

— deplasarea la poziția cursorului a întregului articol cu o poziție spre stînga, realizînd astfel ștergerea caracterului de pe poziția cursorului ;

— deplasarea cursorului de pe poziția curentă la începutul cîmpului următor.

Structura programului de validare și tipurile de validări standard

Programul de validare are o structură liniară, fiind executat instrucțiune după instrucțiune.

Fiecare instrucțiune are două părți :

— *partea fixă* (antet) format din 6 caractere avînd următoarea semnificație :

— două cifre reprezentînd numărul de ordine al cîmpului din articol :

— două cifre reprezentînd lungimea cîmpului ;

— un caracter reprezentînd tipul instrucțiunii numit mnemonică ;

— un caracter „:“ delimitator.

— *partea variabilă* a cărei structură depinde de mnemonica instrucțiunii, conține parametri de validare.

Instrucțiunile standard implementate în sistemul SCVD pot fi clasificate după următoarele criterii :

a) După numărul de cîmpuri luate în considerație :

— *instrucțiuni locale* — care validează un singur cîmp ;

— *instrucțiuni globale* — care validează relațiile între conținutul unor cîmpuri și pozițiile altor cîmpuri.

b) După relațiile cantitative dintre cîmpuri :

— *instrucțiuni condiționale* — validarea unui cîmp se face în funcție de conținutul altor cîmpuri ;

— *instrucțiuni necondiționale* — validarea cîmpului se face numai în funcție de conținutul său.

Cu ajutorul setului de instrucțiuni standard se poate obține un număr mare de variante de validări :

— testarea apartenenței conținutului cîmpului la o listă de variante definită în instrucțiune ;

— testarea conținutului cîmpului ca fiind numeric, alfabetic, alfanumeric sau numai blank-uri. De asemenea, se testează poziționarea informației compacte aliniate la stînga, la dreapta sau într-o poziție intermediară ;

— testarea cîmpurilor numerice, a cărui conținut este cuprins între două limite definite în instrucțiune ;

— testarea cîmpurilor de tip „data zilei" în relație cu data curentă ;

— testarea conținutului unui cîmp, în funcție de conținutul altor cîmpuri ;

— testarea unor relații de tip "<",">", "=" între conținutul numeric al cîmpului curent și conținutul altor cîmpuri numerice legate prin operații de adunare și scădere ;

— transferul într-un cîmp a unei informații fixe definite prin instrucțiune ;

— trecerea deliberată la validarea următorului cîmp, atunci cînd un cîmp nu necesită validare.

Acest set de instrucțiuni poate fi îmbogățit în funcție de necesitățile utilizatorului prin implementarea de către utilizator a unor instrucțiuni proprii. Instrucțiunile utilizatorului trebuie să aibă partea fixă la fel ca instrucțiunile standard, iar partea variabilă în funcție de necesitățile instrucțiunii.

Utilizatorul va scrie o secvență de program care să realizeze testarea sintactică și semantică a propriei instrucțiuni și o altă secvență de program care să realizeze efectiv validarea cîmpului.

Sistemul SCVD pune la dispoziția utilizatorului puncte de intrare, cît și modul de transfer a parametrilor.

Sistemul SCVD este informat de utilizarea noilor instrucțiuni prin funcția „EXTENSIE VALIDARE", realizînd interfața cu utilizatorul.

Concluzii

Sistemul SCVD permite utilizarea microcalculatorului FELIX M-18 pentru a realiza partea de introducere date în cadrul unor sisteme informatice complexe. Avantajele utilizării sistemului sînt în primul rînd de natură economică, eliminînd folosirea suporturilor de informații nereutilizabile cît și a unor activități intermediare.

Sistemul preia o parte din lucrările executate pe un calculator de capacitate mai mare, realizînd o disponibilitate de resurse și timp, ce pot fi folosite în alte scopuri.

9.9. Sistem de dezvoltare pentru studiul structurilor multiprocesor bazat pe module — URC-M118

9.9.1. Generalități

Sistemele de conducere a proceselor industriale beneficiază de avantajele oferite de arhitecturile multiprocesor pentru a rezolva o serie de probleme fundamentale legate de: modularizare, un cost cît mai redus, evoluția incrementală a algoritmilor și a funcțiunilor, adaptabilitatea la o gamă largă de aplicații, fiabilitate cît mai mare etc.

Arhitecturile bazate pe microprocesoare, orientate pe 8 sau 16 biți devin din ce în ce mai răspîndite, datorită faptului că ele pot fi interconectate în conformitate cu cerințele aplicațiilor concrete.

Microprocesoarele curente nu dispun de o memorie proprie în care să se poată păstra programul de calcul.

Astfel, configurațiile multimicroprocesor vor fi organizate în cadrul unei memorii comune în care vor fi stocate programele de lucru și informațiile care urmează a fi prelucrate, respectiv rezultatele prelucrării. În acest caz memoria reprezintă o resursă comună care este adresată de microprocesoare pentru citirea instrucțiunilor și a datelor în vederea prelucrării lor.

Frecvența de acces la memorie depinde de numărul de microprocesoare. Performanțele unui asemenea sistem sînt condiționate de ciclul de lucru al memoriei și de numărul de microprocesoare. Se constată că, din punct de vedere al numărului de operații executate în unitatea de timp, performanțele nu cresc proporțional cu numărul de procesoare din sistem, observîndu-se o tendință de saturație începînd de la valori relativ reduse ale numărului de procesoare.

9.9.2. Structura generală a sistemului de dezvoltare

În realizarea sistemului de dezvoltare s-a plecat de la unitățile de repertoriu comercial pentru familia de microcalculatoare FELIX M 18/118, la care s-au adăugat module funcționale special proiectate și realizate, cum ar fi modulul de cuplare la proces și modulul unității de înmulțire/împărțire în virgulă fixă.

Deoarece microprocesorul 8080 nu posedă memorie proprie, pentru realizarea unei structuri multimicroprocesor cu memorie locală și cu acces la o memorie comună s-a ales o structură de tip microcalculator pe o plachetă, asigurîndu-se în acest fel facilitățile de calcul dorite, și o memorie extensibilă în incremenți de 8 Kocteți.

Tehnica de alocare a procesoarelor pentru prelucrarea proceselor, precum și partajarea procesoarelor la memoria comună vor fi abordate în cadrul aplicației.

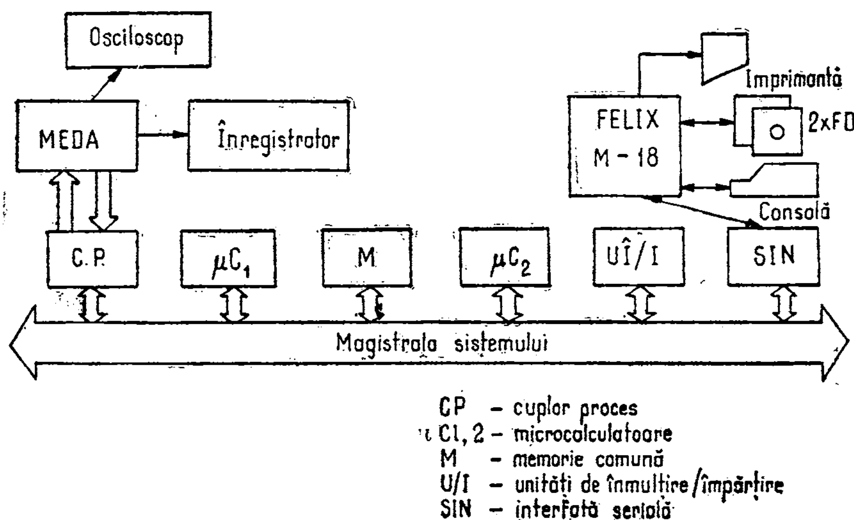


Fig. 9.50. Structura generală a sistemului de dezvoltare.

Structura generală a sistemului de dezvoltare este prezentată în figura 9.50.

Sistemul de dezvoltare conţine următoarele echipamente :

a) sertar FELIX M 118 prevăzut cu sursă de alimentare şi următoarele module funcţionale :

— unităţi de prelucrare (μC_1, μC_2 — microcalculatoare pe o plachetă), care dispun de : memorie locală (8 Kocteţi REPROM, 2 Koct. RAM, sistem de întreruperi, logica de acces la magistrală ;

— unitate de comandă şi sincronizare pentru magistrală ;

— unitate de memorie, extensibilă în incremenţi de 8 Kocteţi (memoria comună) ;

— unitate de cuplare serială SIN ;

— unitate de cuplare la proces ;

— unitate de înmulţire/împărţire în virgulă fixă.

Modulul interfeţei seriale (SIN) este realizat pe o plachetă prevăzută cu un circuit 8251, reprezentînd interfaţa programabilă serială. Viteza de transmisie poate fi modificată prin selecţie manuală în conformitate cu standardele în vigoare, începînd cu 110 b/s pînă la 4 800 b/s. Cupla este compatibilă cu standardul RS-232-C.

Modulul SIN serveşte la cuplarea cu un calculator ierarhic superior sau cu o consolă operator.

b) Calculator FELIX M 18 echipat cu memorie de 64 Kocteţi, unitate duală de disc flexibil, imprimantă şi display. Calculatorul FELIX M 18 se cuplează prin interfaţa serială (SIN) cu sertarul FELIX M 118.

c) Calculator analogic MEDA 42 T, folosit pentru simularea procesului care urmează a fi condus cu sistemul multimicrocalculator.

9.9.3. Descrierea structurii hardware a sistemului de dezvoltare

Programele se elaborează pe microcalculatorul FELIX M 18, sub sistemul de operare SFDX-18. Ele pot fi scrise în limbaj de asamblare (ASM 80) sau în limbaje de nivel înalt (PLM 80, FOR 80). Folosind mijloacele de depanare existente în utilitarele FELIX M-18 se pot verifica și corecta programele de sistem și aplicații pentru microcalculatoarele μC_1, μC_2.

Folosind consola microcalculatorului FELIX M-18 se poate dialoga cu monitorul plasat în μC_1, μC_2, în sensul de a se afișa conținutul unor zone de memorie din memoria locală, de a se citi sau scrie fișiere pe discurile flexibile (FD).

În continuare se descrie modulul de cuplare la proces (CP), prezentat în figura 9.51.

Se pot culege de la proces prin multiplexorul analogic 8 mărimi (intrări), selectate cu liniile de adresă ADR1÷ADR3. Fiecare mărime analogică este convertită în mărime numerică de convertorul A/N, care lucrează pe 12 biți. Deoarece magistrala de date a sistemului este de 8 biți este necesar ca informația să fie preluată la două momente de timp diferite și se depune în perechea de registre generale (D, E).

Deoarece convertorul A/N lucrează ca memorie (la adresele 0EFE0-0EFEFH) poate fi apelat de oricare din cele două microcalculatoare.

Cînd unul din cele două microcalculatoare (de ex. μC_1) citește date de la convertor, ocupă magistrala sistemului, blocînd în acest fel accesul celuilalt microcalculator.

Datele prelucrate în microcalculatorul μC_2 sînt transmise la convertoarele N/A, cuplate de asemenea ca memorii (la adresele 0EFE8H-0EFFBH). Cele 2 convertoare N/A lucrează pe 10 biți. Deoarece convertoarele nu posedă decodificator de adrese (cum au memoriile) este necesar să se decodifice liniile de adresă în exterior, după cum se observă în figura 9.51.

Trebuie remarcat faptul că atît convertorul de intrare (A/N) cît și cele de ieșire (N/A) lucrează în paralel cu memoria microcalculatorului;

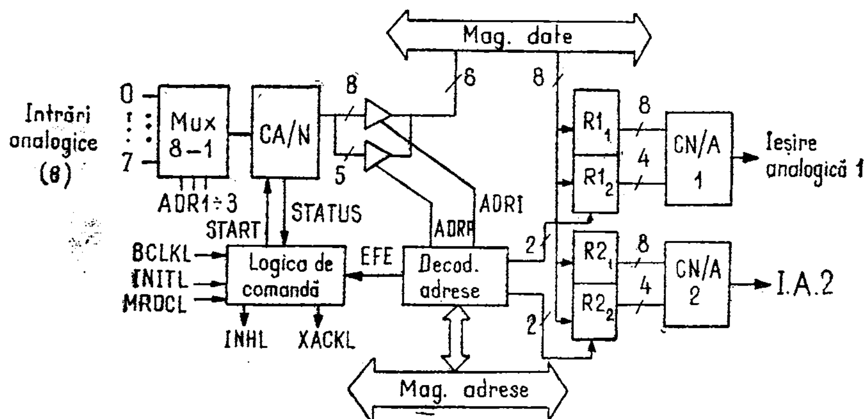


Fig. 9.51. Modulul de cuplare la proces.

Nr canal intrare	Adresă primii 4 biți (MSB)	Adresă următorii 8 biți (LSB)
CH1	0EFE0H	0EFE1H
CH2	0EFE2H	0EFE3H
CH3	0EFE4H	0EFE5H
CH4	0EFE6H	0EFE7H
CH5	0EFE8H	0EFE9H
CH6	0EFEAH	0EFEBH
CH7	0EFECH	0EFEDH
CH8	0EFEEH	0EFEFH

Fig. 9.52. Adresele de memorie ale celor 8 canale analogice.

aceasta impune furnizarea unui semnal de inhibare a memoriei microcalculatorului în momentul citirii de la / scrierii la convertoare.

Selecția unui canal de intrare (din cele 8 canale CH1-CH8) se face cu ajutorul biților de adresă ADR1÷ADR3 ; deci fiecare convertor va ocupa două adrese de memorie : — de la prima adresă (pară, de exemplu pentru canalul 1-0EFE0H) se citesc primii patru biți, începînd cu cel mai semnificativ (MSB) și bitul de stare (STATUS), iar de la cea de-a doua adresă (impară, de exemplu, pentru canalul 1:0EFE1H) următorii 8 biți ai valorii convertite. Adresele celor opt canale de intrare sînt prezentate în tabelul din figura 9.52.

Convertorul analog/numeric utilizat se alimentează cu +15 V, —15 V, +5 V și OV și suportă intrări analogice cuprinse în gama —5 V÷+5 V. Convertorul trebuie să primească semnalul START CONV., necesar pentru începerea conversiei, iar la terminarea conversiei se obține un semnal 1 pe bitul de stare (STATUS), însemnînd că cei 12 biți de informație sînt disponibili utilizatorului.

Cele două convertoare de ieșire N/A se alimentează cu tensiunile +15 V, —15 V și masă și pot furniza ieșiri în domeniul —5 V÷+5 V. Întrucît ele lucrează pe 10 biți puterea lor de rezoluție este de $1/2^{10}$ din 10 volți, deci aproximativ 10 mV.

„Latch"-urile de intrare (în convertoarele N/A) funcționează ca registre de deplasare de o formă mai particulară. Convertorul primește 10 biți de informație, însă placa are doar 8 intrări de date. Primul tact, adică semnalul aplicat de generarea adreselor 0EFF8H, respectiv 0EFFAH încarcă în latch-uri 6 biți de informație cu semnificația 2^5—2^0 (D2—D7) din cei 10 biți care se convertesc. În acest tact informația nu e convertită, intră doar în latch.

Cel de-al doilea tact, rezultat din decodificarea adreselor 0EFF9H și respectiv 0EFFBH, va încărca în latch-urile tampon ale convertoarelor următorii 4 biți de informație (D0—D3) cu semnificațiile 2^9—2^6. Biții înscriși în latch la tactul anterior sînt transferați acum în latch-urile tampon și tot cuvîntul de informație se aplică convertorului.

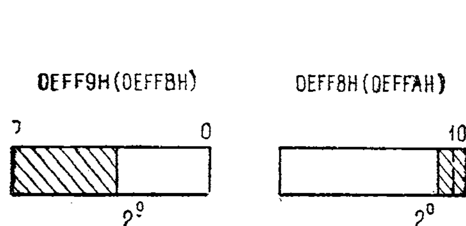


Fig. 9.53. Formatul datelor pentru convertoarele N/A.

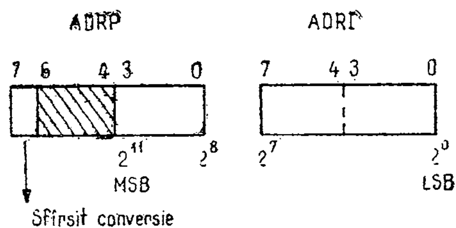


Fig. 9.54. Formatul datelor citite de la convertorul A/N.

Trebuie să se ţină cont de aceste semnificaţii ale biţilor de informaţie ce alcătuiesc cuvîntul de convertit, şi să se trimită spre convertoare informaţia sub forma prezentată în figura 9.53.

Informaţia de la convertorul analog/numeric se citeşte în formatul descris în figura 9.54, unde ADRP — reprezintă prima adresă (pară) a unuia din cele opt canale, iar ADRI — adresa următoare (impară) a aceluiaşi canal.

Pentru convertorul analog/numeric modul de lucru este următorul:

— în momentul decodificării adresei unuia din cele 8 canale de intrare (semnalul EFE) şi a unei operaţii de citire din memorie de la una din aceste adrese (MRDCL) se furnizează convertorului semnalul START CONV.; memoria, cu aceleaşi adrese cu ale convertorului este inhibată (INHL) pentru a nu furniza semnalul de răspuns XACKL.

— semnalul de răspuns XACKL de la modulul de cuplare la proces (datele sînt stabile pe magistrală) va fi furnizat după trecerea semnalului STATUS (de la convertor A/N) din 0 în 1, însemnînd sfîrşitul conversiei şi deci că informaţia este disponibilă utilizatorului.

Unitatea de înmulţire/împărţire în virgulă fixă este concepută ca un echipament periferic, interfaţat cu microcalculatorul FELIX M18 prin porturile de intrare/ieşire ale acestuia.

Unitatea conţine :

— 5 registre (A, B, C, D, E) de 8 biţi fiecare, folosiţi şi ca registre pereche, pentru rezultatele celor două operaţii, cu funcţii diferite pentru înmulţire şi împărţire ;

— unitate aritmetică-logică capabilă să realizeze adunarea şi scăderea pe 8 biţi ;

— blocul de comandă a secvenţei operaţiilor ;

— schemă de reset general înainte de iniţierea oricărei operaţii de înmulţire sau împărţire.

Unitatea are alocate un număr de 7 porturi :

3 de intrare şi 4 de ieşire.

Primul port de ieşire (OUT 10H) este asociat operaţiei de reset general. Următoarele două porturi de ieşire, OUT 11H şi OUT 13H, sînt folosite pentru încărcarea primului şi celui de-al doilea operand concomitent cu operaţia de înmulţire/împărţire propriu-zisă, şi respectiv, la resetarea acesteia după fiecare din operaţiile de încărcare.

Al patrulea port de ieşire, OUT 12H este folosit pentru trimiterea cuvîntului de comandă către unitatea în virgulă fixă; în acumulator se

încarcă în prealabil cuvîntul de comandă de 8 biți, din care sînt utilizați numai ultimii 2 biți, după cum urmează :

A_1	A_0	
0	0	— se trimite spre unitate primul operand pentru înmulțire (deînmulțitul) ;
0	1	— se trimite al doilea operand la înmulțire și împărțire ;
1	0	— se trimite primul operand pentru împărțire (deîmpărțitul) ;
1	1	— se solicită preluarea rezultatului.

Porturile de intrare sînt utilizate astfel :

IN	10H	— testarea prin program a sfîrșitului operației ;
IN	11H	— citirea cuvîntului de stare al unității (unitatea nu necesită teste de depășire superioară sau inferioară a rezultatului, ci doar test de împărțire prin zero, în care caz furnizează 0FFH) ;
IN	12H	— acest port este folosit de două ori succesiv pentru citirea celor doi octeți ai rezultatului înmulțirii sau împărțirii.

Rezultatul înmulțirii se obține pe 16 biți și va fi preluat de sistemul de calcul prin două comenzi succesive (IN 12H), care permit citirea întîi a octetului mai semnificativ și apoi a celui mai puțin semnificativ. Rezultatul împărțirii se obține în mod asemănător, cu specificarea că primul octet reprezintă partea întreagă (cîtul), iar cel de-al doilea partea fracționară (restul).

Numerele se consideră reprezentate în cod complementar.

Exemplu de utilizare a unității în virgulă fixă pentru efectuarea unei înmulțiri :

```
        OUT    10H      ; reset general
        MVI    A, 00H   ; codul operației+primul operand
        OUT    12H
        MVI    A, OP1   ; se încarcă primul operand
        OUT    11H      ; generarea și resetarea semnalului
        OUT    13H      ; de ceas după trimiterea operandului
        MVI    A,01H    ; codul de operare pentru al
        OUT    12H      ; doilea operand
        MVI    A,OP2    ; se încarcă al doilea operand
        OUT    11H
        OUT    13H
        OUT    11H      ; începerea operației propriu-zise
TEST :  IN     10H      ; secvența de așteptare în
        RAL
        JC     TEST     ; buclă a rezultatului
        OUT    13H      ; resetarea ceasului
        MVI    A, 03H   ; codul de operare pentru
        OUT    12H      ; preluarea rezultatului
        IN     12H      ; preluarea octetului mai semnificativ
        CMA             ; al rezultatului și depunerea lui
        MOV    B, A     ; în registrul B
        OUT    13H
```

```
IN      12H     ; secvenţa de preluare a octetului
CMA
MOV     C, A    ; mai puţin semnificativ al rezultatului
OUT     13H     ; şi plasarea lui în registrul C.
```

Utilizînd unitatea în virgulă fixă, timpul de execuţie al unei operaţii de înmulţire/împărţire este de aproximativ 120 μs.

9.9.4. Operaţii software de bază în sisteme multimicro

Pentru cele două microcalculatoare μC_1, μC_2 a fost elaborat un monitor rezident simplificat care permite efectuarea unui dialog între aceste microcalculatoare pe o plachetă şi sistemul FELIX M18. Astfel, folosind consola microcalculatorului FELIX M18 se poate dialoga cu monitorul plasat în μC_1, μC_2 în sensul de a afişa conţinutul unor zone de memorie din memoria locală, de a se citi sau scrie fişiere pe unităţile de discuri flexibile (FD).

Monitorul cuprinde următoarele comenzi :

>L <nume fişier> ; încarcă de pe unitatea 0 fişierul cu numele specificat.

>T <adresa 1>, <adresa 2>, <nume fişier> ; transferă conţinutul zonei de memorie cuprinsă între cele două adrese pe unitatea 0, asociindu-i numele specificat.

>D <adresa 1>, <adresa 2> ; afişează la consolă zona de memorie cuprinsă între cele 2 adrese.

>X ; afişează la consolă starea programului întrerupt.

>G <adresa 1>, <adresa 2>, <adresa 3> ; lansează în execuţie un program începînd de la prima adresă, cu posibilitatea de suspendare cînd este atinsă una din următoarele două adrese.

>F <adresa 1>, <adresa 2>, <parametru> ; se încarcă zona de memorie specificată cu un parametru.

Comunicaţia între microcalculatoarele μC_1 şi μC_2 se efectuează folosind memoria comună. Deoarece semaforul S, asociat zonei tampon din memorie, prin intermediul căreia se face schimbul de date, este plasat într-o celulă a aceleiaşi memorii, manipularea lui trebuie făcută printr-o operaţie indivizibilă (citeşte-modifică-înscrie). Pentru a realiza acest deziderat, în cazul microprocesorului 8080, trebuie cîştigat accesul la magistrală şi menţinut accesul pe durata manipulării semaforului.

Pentru această operaţie se pot folosi primitivele P şi V ; ca proceduri scrise în limbajul PL/M vor avea următoarea structură :

```
DECALARE
S$ADR ADDRESS ;
DECLARE BUS$LOCK LITERALLY '0FEH',
            LOCK LITERALLY '0', UNLOCK LITERALLY '1' ;
V :
PROCEDURE (S$ADR) ;
DECLARE (S BASED S$ADR) BYTE ;
```

```
OUTPUT (BUS$LOCK) = LOCK ;
S=S+1 ;
OUPUT (BUS$LOCK) = UNLOCK ;
END V ;
P : PROCEDURE (S$ADR) ;
  DECLARE (S BASED S$ADR) BYTE ;
    DO FOREVER
      IF S>0 THEN
        DO ;
        OUTPUT (BUS$LOCK) = LOCK ;
          IF S>0 THEN
            DO ;
              S=S—1
                OUTPUT (BUS$LOCK) = UNLOCK ;
              RETURN ;
            END ;
          OUTPUT (BUS$LOCK) = UNLOCK ;
        END ;
      END ;
    END P ;
```

Comunicația între procese implică relații de tip producător-consumator. Datele produse sînt plasate într-o zonă tampon de memorie. Din această zonă datele sînt consumate de către un alt proces.

Pentru implementarea acestui tip de comunicație se vor folosi semafoare : EMPTY, FULL și SEMA.

Semaforul EMPTY este egal cu numărul de celule vide din zona tampon (inițial este egal cu numărul de buffere din zonă — NMB$BUFFERS).

Semaforul FULL specifică numărul de celule ocupate din zona tampon (inițial este 0).

Semaforul SEMA controlează accesul propriu-zis la zona tampon în legătură cu plasarea sau extragerea datelor din ea, pentru ca aceste operații să aibe un caracter secvențial.

În figura 9.55 este dată organigrama comunicației producător-consumator.

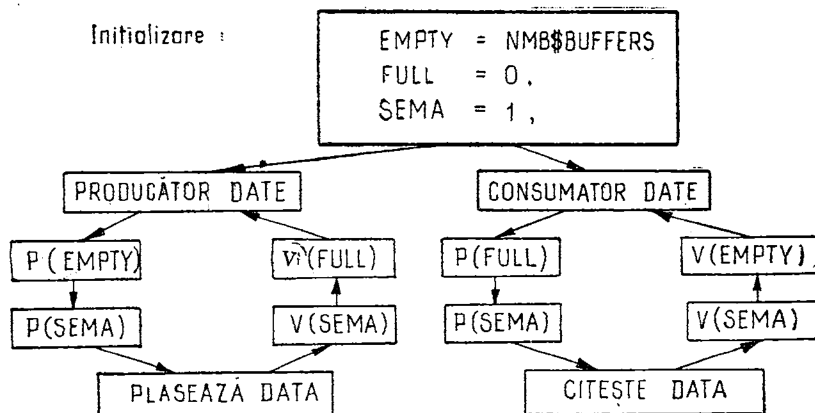


Fig. 9.55. Organigrama modelului producător-consumator.

Dacă EMPTY=0, producătorul aşteaptă, iar dacă FULL=0, consumatorul aşteaptă.

În mod similar se poate trata utilizarea unei resurse comune cum ar fi modulul înmulţire/împărţire.

9.9.5. Exemplu de folosire a unei structuri biprocesor destinată conducerii unui proces industrial simulat pe calculatorul analogic MEDA

Pentru exemplificare se poate considera procesul simulat pe calculatorul analogic MEDA. Mărimile analogice din proces sînt preluate de către microcalculatorul μC_1 folosind modulul cuplor de proces. Datele sînt comparate cu limitele admise, filtrate şi plasate într-o zonă tampon, de intrare, din memoria comună.

Microcalculatorul μC_2 preia aceste date şi le prelucrează în conformitate cu un algoritm dat, folosind pentru operaţiile de înmulţire/împărţire modulul specializat „slave“ conectat pe magistrală. Rezultatele sînt plasate într-o zonă tampon de ieşire, în memorie, de unde sînt preluate de microcalculatorul μC_1. Acesta le transmite spre proces folosind convertoarele numeric-analogice din modulul cuplor de proces.

Întrucît μC_1 dispune (ca urmare a unei analize a timpului de execuţie) de o rezervă de timp, el va prelua şi funcţiunile de comunicaţie cu calculatorul ierarhic superior (FELIX M18), respectiv cu operatorul.

Procesul analogic simulat pe calculatorul MEDA 42T are următoarea funcţie de transfer :

$$H(s)=\frac{0{,}25}{(10s+1)(2s+1)}\,.$$

Schema bloc a întregului sistem implementat (incluzînd reacţia şi blocul regulator) este prezentată în figura 9.56.

Mărimile indicate pe figură au următoarea semnificaţie :

y(t) — mărimea de ieşire din proces (simulat pe MEDA)-continuă ;
u(t) — mărimea de comandă continuă, respectiv ieşirea convertorului N/A ;
y(k) — ieşirea discretizată a procesului, respectiv convertită numeric de convertorul A/N ;

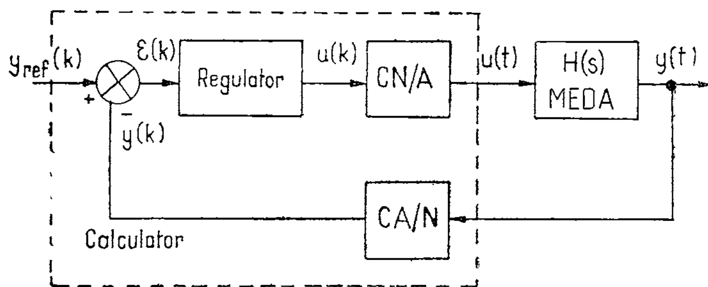


Fig. 9.56. Schema bloc a sistemului implementat.

u(k) — mărimea de comandă discretizată, respectiv ieşirea din calculator la pasul k ;

$\varepsilon(k)=y_{ref}(k)-y(k)$ eroarea calculată la pasul k ;

$t=kT$, unde T — perioada de eşantionare.

Pentru o funcţie de ordinul II de forma :

$$H(s)=\frac{k}{(1+T_1s)(1+T_2s)},$$

transformata z are următoarea formă :

$$H(z)=\frac{k}{T_1-T_2}\cdot\frac{(T_1b_1-T_2b_2)\,z+(b_2a_1T_2-b_1a_2T_1)}{(z-a_1)(z-a_2)};$$

$a_1=e^{-\frac{T}{T_1}}$; $a_2=e^{-\frac{T}{T_2}}$;

$b_1=1-a_1$; $b_2=1-a_2$.

Compensatorul (regulatorul) pentru această funcţie are forma :

$$H_c(z)\stackrel{\Delta}{=}\frac{U(z)}{\varepsilon(z)}=k_c\,\frac{z-a_1}{z-1}.$$

Alegînd perioada de eşantionare T=50 ms se obţine pentru instalaţia considerată, următorul regulator :

$$H_c(z)=0{,}25\,\frac{z-0{,}95}{z-1}=0{,}25+\frac{0{,}0125}{z-1}.$$

Discretizînd se obţine :

$X_c(k+1)=X_c(k)+0{,}0125\,\varepsilon(k)$

$U(k)=X_c(k)+0{,}25\,\varepsilon(k)$

Implementînd legea de reglare discretizată, pe μC_2, în limbaj de nivel înalt (de ex. PL/M) se constată că durata de calcul a comenzii la pasul k este de aproximativ 10 ms (pentru operaţiile aritmetice s-a utilizat biblioteca aritmetică în virgulă mobilă FPAL).

Aceleaşi operaţii efectuate prin folosirea modulului „slave" de înmulţire/împărţire în virgulă fixă, sînt realizate într-un timp cu un ordin de mărime mai mic.

De notat faptul că datorită componentelor fizice cu care a fost implementat cuplorul de proces, apar restricţii în ceea ce priveşte conducerea proceselor rapide.

Din analizele efectuate şi măsurătorile asupra cuplorului de proces se constată că timpul necesar citirii valorilor analogice, conversiei, interpretării şi generării răspunsului este de minimum 200 µs.

Sistemul proiectat şi realizat destinat conducerii proceselor industriale poate fi utilizat la procese pentru care variaţiile mărimilor nu depăşesc 5 kHz.

9.10. Utilizarea calculatorului Felix M-18 în echipamente de testare automată

9.10.1. Testoarele automate THETA

Echipamentele de testare automată sînt o replică firească a dezvoltării fără precedent a electronicii în ultimii ani, a creşterii continue a gradului de integrare al circuitelor electronice şi a realizării de echipamente complexe extrem de compacte, specifice tehnicii numerice.

Acest fapt a impus verificarea echipamentelor realizate în această tehnologie compactă, atît în cursul fabricării, cît şi al detectării defectelor şi reparării în exploatare. Automatizarea proceselor de verificare şi control, atît în fluxurile de fabricaţie cît şi în activitatea de service [21], este cunoscută sub termenul de TESTARE AUTOMATĂ, iar echipamentele respective ca ECHIPAMENTE DE TESTARE AUTOMATĂ — ETA; în lume l-i se acordă astăzi o importanţă, corelată cu efortul general depus în dezvoltarea ETA în toate domeniile de fabricaţie industrială. La acest efort general s-a aliniat şi ţara noastră, prin contribuţia adusă de institutele de cercetare ştiinţifică, institutele de învăţămînt superior şi întreprinderile industriale. În cele ce urmează se vor face referiri numai la ETA realizate la IPA filiala Cluj Napoca [1], preponderent dotate cu calculatorul M-18, şi realizate, îndeosebi, pentru testarea plachetelor echipate.

Echipamentele cunoscute sub denumirea „THETA", reprezintă o prescurtare a expresiei „Tehnologii şi Echipamente de Testare Automată", însoţite de un cod. Se cunoaşte astfel seria THETA FD 5000, privind domeniul testării funcţional dinamice a plachetelor echipate şi anume: 5 010 pentru testări numerice, 5 020 pentru testări analogice, 5 030 pentru testări de sisteme cu microprocesoare, 5 050 pentru testări hibride. Aceste echipamente permit testări de tipul „funcţional dinamic", respectiv verifică placheta în condiţii cît mai apropiate de funcţionarea reală a acesteia. Această serie a fost întregită cu echipamentul THETA IN 4010, care permite verificarea plachetelor după tehnica „în-circuit", respectiv o testare din punctul de vedere al corectitudinii echipării plachetei cu componente active şi pasive.

Structural, ETA se compune dintr-un calculator, în cazul de faţă M-18, dotat cu un număr de periferice standard, dar minimal cu un floppy-disk dual şi un display DAF 2010. Celelalte periferice, ca imprimantă, centronix, cititor/perforator de bandă, cititor de cartele sînt opţionale şi pot întregi configuraţia, în special în raport cu cerinţele beneficiarului de a dezvolta programe de test evoluate. Echipamentul de test propriu-zis poate fi privit ca o configuraţie de module „Slave", care lucrează sub controlul calculatorului, ca un sistem. Performanţele unui ETA depind, în egală măsură şi de software-ul specific, astfel încît alegerea unui calculator pentru un echipament de test nu este o chestiune nici simplă şi nici unică.

[1] M. Hănganuţ, dr. ing., este profesor universitar la Institutul Politehnic Cluj Napoca şi şeful filialei de testare automată a Institutului de Cercetare Ştiinţifică şi Inginerie Tehnologică pentru Automatizări (IPA) din Cluj-Napoca. A. Predoi, este Directorul IPA.

9.10.2. Atribute hardware ale lui Felix M-18 pentru testoare

Asimilarea și dezvoltarea calculatorului Felix M-18 a mers aproape paralel cu elaborarea ETA la IPA Cluj Napoca, permițînd de la început atacarea domeniului echipamentelor evoluate, similare ca performanțe cu cele realizate pe plan mondial de către firme specializate.

O primă problemă în utilizarea calculatorului M-18 în ETA constă în posibilitatea extinderii bus-ului propriu, prin existența plachetei bus, care pune la dispoziție porțiunile de input/output necesare schimbului de informații între calculator și testorul propriu-zis. La ETA din seria 5 000 s-au utilizat diferite moduri de cuplare, ca interfațare directă tip daisy chain, interfațare printr-un controller microprogramat și o magistrală a sistemului de test, interfațare prin controller IEC 625 (IEEE 488) și bus aparținînd aceluiași standard. Dintre aceste soluții, care au prezentat fiecare avantaje și dezavantaje proprii, ultima soluție pare a fi cea mai avantajoasă și a fost, în consecință, adoptată, inclusiv pentru „Sistemul Unic de Testare Automată" — SUTA, în curs de elaborare ca modul de referință pentru țările membre CAER, cu o interfațare CAMAC pentru a permite utilizarea acestor module în echipamentele de testare.

O altă facilitate oferită de calculatorul M-18 și utilizată în ETA este sistemul de întreruperi, care permite o eficientă conlucrare între unitatea centrală și modulele de test, asigurînd cvasi-simultanietatea diverselor aplicații și mărind viteza de lucru generală a sistemului. Permite, de asemenea, tratarea unor situații de excepție, ce pot apărea în exploatarea ETA, ca și — împreună cu ceasul de timp real — efectuarea unor verificări periodice asupra unor variabile critice dintr-un anume program de test. De asemenea, în ETA este utilizat accesul direct la memorie (DMA), care permite transferul blocurilor masive de date, așa cum apar, de ex. frecvent, la testări dinamice.

În sfîrșit, dotarea ultimelor calculatoare ale familiei M-18 cu multiplexor, permite realizarea de ETA în configurații distribuite și ierarhice, ceea ce reprezintă o tendință actuală în testarea automată, anume aceea de a crea sisteme de testare integrate, în conceptul mai general CADMATR (Computer aided design, manufactory, automatic testing and repaire).

9.10.3. Dezvoltări de software pentru testare

Dintre facilitățile software oferite de calculatorul M-18, sistemul de operare SFDX-18 stă la baza dezvoltării software-ului de testare specific. Astfel, la IPA Cluj-Napoca a fost elaborat un limbaj orientat spre testare LITEST, utilizat în prezent în trei variante: LITEST 1 — limbaj de nivel mediu specific ETA pentru plachete numerice, în special în variante constructive „mini" : LITEST 2 — limbaj de nivel înalt derivat din limbajul ATLAS internațional, adoptat și utilizat la ETA pentru plachete hibride; LITSET 2E — prezintă o completare a limbajului LITEST 2 cu facilități pentru tehnica testării „în-circuit". Toate aceste limbaje oferă utilizatorului multiple posibilități de elaborare a programelor de test, cu un nivel aprofundat de diagnoză.

Pentru ușurarea elaborării programelor de test s-a dezvoltat, de asemenea, un generator interactiv GINTA, care permite lucrul ETA în dialog cu aplicatorul. În curs de elaborare se află și generatoare automate de programe. S-a dezvoltat, de asemenea, tehnica programării prin învățare, utilizînd o plachetă cunoscută ca bună.

ETA din seria 5 030 au necesitat dezvoltarea unui software specific, propriu modului de lucru al acestor echipamente, prin tehnica emulării în circuit şi a analizei de semnătură. Limbajele de testare implementate în echipamentele THETA dispun de translatoare, care lucrează sub controlul sistemului de operare SFDX-18, permiţînd translatarea programului sursă în cod obiect abstract, executabil prin intermediul procesorului „executiv" specific fiecărui ETA. La toate aceste elaborări software s-a avut în vedere o concepţie structurală, care oferă o mare flexibilitate şi adaptare la particularităţile diferitelor echipamente din formula THETA.

De un real folos în special în elaborarea softwareului specific de testare sînt şi compilatoarele FORTRAN, BASIC G-18, PASCAL, rezidente în calculatorul M-18, cît şi biblioteca aritmetică frecvent utilizată, în special la testările analogice.

9.10.4. Limitări ale lui Felix M-18 în testarea automată

Cu toate aceste avantaje, care au scos în evidenţă concepţia evoluată a calculatorului M-18 şi, mai ales, posibilitatea utilizării eficiente a acestuia în construcţia ETA, unele limitări inerente modului în care M-18 a fost conceput şi realizat îl fac utilizabil doar la ETA de nivel mediu al clasei superioare. Aceste limitări se datoresc, în special, memoriei de 64 K, insuficiente pentru teste de mare complexitate. Existenţa unui modul de virgulă mobilă prin hardware permite substanţial creşterea vitezei de lucru în anumite aplicaţii, ca şi dezvoltarea ceasului de timp real şi a sistemului de întreruperi, care vor asigura aplicaţii de timp mai evoluate şi rezolvate într-un mod mai uniform.

9.10.5. Caracterizări ale noilor testoare THETA

Echipamentele de testare automată din seria FD 5 000 sînt realizate modular astfel încît cu un număr redus de module să se poată obţine configuraţii diferite, perfect adaptabile cerinţelor utilizatorului şi să evite costurile testării, care uneori depăşesc pe cele productive [22].

Prezent în practic toate ETA elaborate pînă acum este modulul numeric, mai recent într-o variantă evoluată. Modulul este interfaţat cu sistemul de calcul, printr-o interfaţă specifică tipului de magistrală testor utilizată recent, IEC 625. Electronica de pin este realizată modular pe plachete echipate standard cu circuitele „driver" şi „sesizor", care permit aplicarea de secvenţe de test formate din 0/1 logic şi tren de impulsuri în familiile logice TTL şi CMOS şi citirea răspunsurilor plachetei. Frecvenţa de lucru este de pînă la 1 MHz în varianta standard şi pînă la 10 MHz în varianta evoluată, cînd modulul este dotat cu memorie de pin. Tot acestui modul îi poate fi ataşat un analizor de semnături mono sau multicanal. Cu acest modul se pot realiza ETA pentru plachete numerice, oferind multiple posibilităţi şi flexibilitate în exploatare.

Echipamentele de testare analogică conţin o gamă variată de module de generare şi măsurare ca : generator de funcţii programabil, frecvenţmetru, numărător cu două canale, voltmetre şi multimetre numerice, inclusiv instrumente interfaţate, surse programabile de tensiune şi curent, detector de fază, amplificatoare instrumentale etc.

Pentru a conferi ETA un caracter cît mai universal și expandabil, acestea sînt echipate cu interfețe IEC 625, care permit conectarea la sistem a oricărui instrument prevăzut cu interfață pentru acest standard.

Pentru testarea plachetelor echipate cu microprocesoare a fost elaborat un modul specific de emulare în circuit, care permite prelucrarea funcțiilor microprocesorului, în prezent I 8 080, în vederea rulării unor programe de test într-un mod flexibil și evoluat. Tot atașat ETA cu această destinație este un analizor de semnături multicanal și un analizor logic.

Echipamentul de tip IN 4 010 mai conține, în plus față de modulul numeric și modulul analogic, un modul specific pentru detectarea scurtcircuitelor și intreruperilor pe plachete neechipate sau echipate. Dacă structural echipamentele din familia FD 5000 și IN 4010 sînt, în bună măsură, asemănătoare, nu același lucru se poate spune despre software care este specific elaborat în special în ceea ce privește programele executive, pentru fiecare tip de echipament în parte.

*

* *

Desigur această prezentare succintă nu a putut nici pe departe pune în evidență trăsăturile și capabilitatea pe care echipamentele din familia THETA le posedă, cît și faptul că testarea automată presupune ca blocurile electronice să fie concepute pentru acest scop [23]. Cititorului interesat i se recomandă consultarea volumelor Primului Simpozion de Tehnologii și Echipamente de Testare Automată ce a avut loc la Cluj-Napoca în noiembrie 1982. Volumele celui de al doilea simpozion, din noiembrie 1983, cît și Volumul AMC — Testare Automată vor apărea în cursul acestui an.

Anexa 2

Mesaje de eroare produse de SFDX

Această anexă listează mesajele de eroare emise de diferitele comenzi ale SFDX-18.

Prin convenţie, numerele 1—99 inclusiv sînt rezervate erorilor detectate de rutinele rezidente sistemului. Erorile numerotate cu 100—199 inclusiv, sînt rezervate pentru programele utilizatorului şi numerele 200—255 inclusiv sînt utilizate pentru erorile care pot fi întîlnite de rutinele nerezidente ale sistemului.

Erorile ale căror numere sînt precedate de asterisc, sînt erori fatale.

Toate celelalte erori sînt în general erori reparabile în afară de cazul în care sînt emise de apelul funcţiei CONSOL.

0 nici o eroare detectată
*1 Spaţiu insuficient în zona de buffere SFDX pentru un buffer cerut
2 AFTN — nu specifică un fişier deschis
3 Încercarea de a deschide mai mult de 6 fişiere simultan
*4 Specificaţie ilegală a unui nume de fişier
*5 Specificaţia dispozitivului în nume fişier este ilegală sau nerecunoscută
6 Încercarea de a scrie într-un fişier deschis pentru intrare
*7 Operaţie abandonată, spaţiu insuficient pe discul flexibil
8 Încercarea de a citi dintr-un fişier deschis pentru ieşire
9 Nu mai este loc în fişierul director
10 Numele fişierelor nu specifică acelaşi disc flexibil
11 Nu se poate redenumi un fişier ; numele este deja utilizat
12 Încercarea de a deschide un fişier deja deschis
*13 Nu există nici un fişier pe disc cu numele specificat în comandă
*14 Încercarea de a deschide pentru scriere (ieşire sau reactualizare) sau a şterge sau a redenumi un fişier protejat la scriere
*15 Încercarea de a încărca în zona SFDX-18 sau în zonele de buffere
*16 Eroare de sumă de control
17 Încercarea de a redenumi sau a şterge un fişier care nu este pe disc
*18 Nerecunoaşterea apelului funcţiei sistem
19 Încercarea de a executa funcţia SEEK într-un fişier care nu este pe disc flexibil
20 Încercarea de a poziţiona înapoi peste începutul fişierului
21 Încercarea de a executa funcţia RESCAN într-un fişier care nu este editat linie cu linie
*22 Parametrul ACCESS ilegal pentru OPEN sau mod de acces imposibil pentru fişierul specificat (de exemplu, citire de la :LP:)
*23 Nu se specifică nume de fişier pentru un fişier pe disc
*24 Eroare de I/E pe disc
25 Specificaţie incorectă a fişierului ecou la apelul funcţiei OPEN
26 Parametrul ATRIB incorect în apelarea funcţiei sistem ATTRIB
27 Parametrul MODE incorect în apelarea funcţiei sistem SEEK
*28 Nu există extensie de fişier

*29 Sfîrşit de fişier la :CI:
*30 Unitate disc nepregătită
31 Încercarea de a căuta într-un fişier deschis pentru ieşire
32 Nu se poate şterge un fişier deschis
*33 Parametru ilegal în apelul funcţiei sistem
34 Parametrul SWITCH nu este corect în LOAD
35 Încercarea de extindere a unui fişier deschis pentru intrare căutînd cu funcţia SEEK după EOF
201 Switch nerecunoscut
202 Caracter delimitator nerecunoscut
203 Sintaxă de comandă incorectă
204 EOF prea tîrziu
206 Etichetă disc ilegală
207 Nu s-a găsit nici o instrucţiune END în fişierul de intrare
208 Eroare la suma de control
209 Secvenţă de înregistrare ilegală în fişierul obiect
210 Memorie insuficientă pentru întreaga lucrare
211 Înregistrare modul obiect prea lungă
212 Tipul înregistrării modulului obiect este eronat
213 Adresare ilegală specificată în fişierul modul obiect
214 Parametru eronat în fişierul SUBMIT
215 Argument prea lung în fişierul SUBMIT
216 Prea mulţi parametri în fişierul SUBMIT
217 Înregistrare modul obiect prea scurtă
218 Formatul înregistrării modulului obiect este ilegal
219 Eroare de fază
220 Lipsa înregistrării EOF în fişierul modul obiect
221 Segmentul depăşeşte 64 Kocteţi
222 Înregistrare nerecunoscută într-un fişier modul obiect
223 Pointer-ul înregistrării de adresare este incorect
224 Secvenţă de înregistrare ilegală în fişierul modul obiect
225 Numele modulului specificat este ilegal
226 Numele modulului depăşeşte 31 caractere
227 Sintaxa comenzii cere paranteză stînga
228 Sintaxa comenzii cere paranteză dreapta
229 Controlul specificat în comandă nu este recunoscut
230 Simbol definit de mai multe ori
231 Fişier deja existent
232 Comandă nerecunoscută
233 Sintaxa comenzii cere o clauză „TO“
234 Nume de fişier duplicat ilegal în comandă
235 Fişierul specificat în comandă nu este un fişier bibliotecă
236 Mai mult de 249 segmente comune în fişierele de intrare
237 Segmentul de comun specificat nu este găsit în fişierul obiect
238 Conţinutul ilegal al înregistrării de tip stivă în fişierul obiect
239 Nu există modul de tip „header“ în fişierul obiect de intrare
240 Programul depăşeşte 64 Kocteţi

Cînd apare eroarea cu numărul 24, este afişat la consolă mesajul adiţional

FDCC = 00nn, DRIVE = mm

unde nn are următoarele semnificaţii :

01 înregistrare ştearsă
02 eroare CRC (în cîmpul de date)
03 marcă de adresare invalidă
04 eroare de căutare
08 eroare de adresare
0A eroare CRC (în cîmpul ID)
0E nu există marcă de adresare
0F marcă de adresare de date incorectă
10 eroarea de ritm a datelor
20 protejat la scriere
40 eroare de scriere
80 disc neoperaţional

Numerele de erori nefatale stabilite de apelurile de funcții sistem

OPEN	3, 4, 5, 9, 12, 13, 14, 22, 23, 25, 28
READ	2, 8
WRITE	2, 6
SEEK	2, 19, 20, 27, 31, 35,
RESCAN	2, 21
CLOSE	2
DELETE	4, 5, 13, 14, 17, 23, 28, 32
RENAME	4, 5, 10, 11, 13, 17, 23, 28
ATTRIB	4, 5, 13, 23, 26, 28
CONSOL	toate erorile sînt fatale
WHOCON	nici una
ERROR	nici una
LOAD	3, 4, 5, 12, 13, 22, 23, 28, 34
EXIT	nici una
SPATH	4, 5, 23, 28

Erorile fatale furnizate de apelurile de funcții sistem

OPEN	1, 7, 24, 30, 33
READ	24, 30, 33
WRITE	7, 24, 30, 33
SEEK	7, 24, 30, 33
RESCAN	33
CLOSE	33
DELETE	1, 24, 30, 33
RENAME	1, 24, 30, 33
ATTRIB	1, 24, 30, 33
CONSOL	1, 4, 5, 12, 13, 14, 22, 23, 24, 28, 30, 33
WHOCON	33
ERROR	33
LOAD	1, 15, 16, 24, 30, 33
SPATH	33

Mesaje de eroare la comanda LINK

Mesaje de erori fatale

Erorile fatale întîlnite de LINK determină apariția mesajelor ce sînt transmise la consolă. LINK termină prelucrarea și cedează controlul lui SFDX-18.

Erorile care sînt cauzate de intrarea într-o comandă improprie sînt urmate de imaginea parțială a comenzii afișînd caracterul (=) în vecinătatea erorii.

— DUPLICATE FILE NAME

<imagine parțială a comenzii>

Același nume de fișier specifică și fișierul de intrare și fișierul de ieșire.

— INSUFFICIENT MEMORY

LINK nu poate termina prelucrarea din cauza memoriei insuficiente pentru spațiul de lucru care este necesară în primul rînd pentru tabela de simboli.

— INVALID MODULE NAME

<imagine parțială a comenzii>

Numele modulului începe cu un caracter ilegal.

— INVALID SYNTAX

<imagine parțială a comenzii>

Este o eroare în comanda în curs de intrare. Eroarea poate să fie un cuvînt cheie nerecunoscut, un caracter diferit de blank care urmează unui & sau absența unui TO după numele fișierelor de intrare.

— <nume fișier>, BAD FIXUP RECORD

Aceasta este o eroare în formatul intern al modulului obiect relocatabil. Această eroare poate fi rezultatul scrierii greşite a numelui fişierului. Eroarea poate avea loc cînd limbajul translator produce modulul, sau într-un LINK anterior, şi în acest caz trebuie translatat din nou modulul sursă şi reluat LINK.

— <nume fişier>, BAD RECORD SEQUENCE

Aceasta este o eroare în formatul intern al fişierului specificat. Această eroare poate fi rezultatul scrierii greşite a numelui fişierului. Eroarea poate avea loc cînd limbajul translator produce modulul sau într-un LINK anterior, şi în acest caz trebuie translatat din nou modulul sursă şi reluat LINK.

— <nume fişier>, SEGMENT TOO LARGE

Segmentul de ieşire este mai mare de 64 Ko. Porţiunea de segment din fişierul specificat în mesaj nu poate fi adăugată la segmentul de ieşire, pentru că el va fi mai mare de 64 Ko.

— LEFT PARANTHESIS EXTECTED

<imagine parţială a comenzii>

Circuitele cheie, PUBLICS, NAME sau PRINT nu sînt urmate de paranteză stînga „(".

— MODULE NAME TOO LONG

<imagine parţială a comenzii>

Numele modulului, are mai mult de 31 caractere.

— RIGHT PARANTHESIS EXPECTED

<imagine parţială a comenzii>

Lista care urmează cuvintelor cheie PUBLICS, NAME sau PRINT nu este terminată cu paranteză dreaptă „)".

— UNRECOGNIZED CONTROL

<imagine parţială a comenzii>

A fost înlocuit un alt şir de caractere decît NAME, MAP sau PRINT cînd era aşteptat un cuvînt cheie de control.

Mesaje de erori nefatale

Erorile nefatale aplicate de LINK sînt scrise în fişierul hartă. LINK termină prelucrarea înainte de a se întoarce în SFDX-18.

— MORE THAN 1 MAIN MODULE

LINK a găsit mai mult de un modul principal în lista de intrare. Toate modulele principale sînt incluse în modulul de ieşire, dar şirul de adrese al modulului de ieşire este luat de la primul modul principal din lista de intrare.

— <nume modul>, MODULE NOT FOUND IN LIBRARY

Modulul din lista de intrare nu a fost găsit în biblioteca specificată.

— <nume>, COMMON/PUBLIC/EXTERNAL NAME CLASH

Segmentul comun are acelaşi nume ca un simbol declarat public sau external.

— <nume>, HAS DIFFERING TYPES

Un simbol extern şi simbolul (declarat public) cu care se leagă nu sînt de acelaşi tip.

— <nume>, MULTIPLE DEFINITION

Două sau mai multe simboluri globale cu acelaşi nume.

— <nume>, UNEQUAL COMMON LENGTHS

Au fost întîlnite două segmente comune cu acelaşi nume dar cu lungimi diferite.

Mesaje de eroare la comanda LOCATE

Mesaje de erori fatale

În cazul unor erori fatale LOCATE termină prelucrarea şi redă controlul lui SFDX-18.

— <nume fişier>, CHECKSUM ERROR

RECORD TYPE xxH, RECORD NUMBER x

Înregistrarea din fişierul de intrare specificată în mesaj conţine eroarea sumei de control (suma de control calculată nu a fost egală cu suma de control specificată).

— <nume common>, COMMON NOT FOUND

Modulul nu conţine segmentul de common care a fost specificat în comandă.

— <nume fişier>, FIXUP BOUNDS ERROR
RECORD TYPE xxH, RECORD NUMBER x

Fişierul conţine o adresă invalidă.

— <nume fişier>, ILLEGAL RELO RECORD
RECORD TYPE xxH, RECORD NUMBER x

S-a detectat fie un tip de înregistrare ilegal, fie o secvenţă de înregistrări ilegală.

— <nume fişier>, ILLEGAL STACK CONTENT RECORD
RECORD TYPE xxH, RECORD NUMBER x

Tentativă ilegală de iniţializare a segmentului stivă.

— <nume fişier>, INSUFFICIENT MEMORY

Nu este loc suficient pentru prelucrare. Fişierul specificat în mesajul de eroare, este fişierul temporar utilizat de LOCATE.

— [entitate] INVALID SYNTAX

Comanda LOCATE are o eroare de sintaxă, [entitate] este inclusă în mesaj dacă eroarea se referă la o entitate.

Eroarea poate fi una din următoarele :

— caracter diferit de blank care urmează după & ;
— definire multiplă a segmentului în controlul ORDER ;
— cuvînt cheie de control ilegal ;
— absenţa unui "/" după un nume de segment de common în controlul ORDER ;
— nume ilegal de segment în controlul ORDER ;
— nume ilegal de modul în controlul NAME.

— <nume fişier>, NO MODULE HEADER RECORD

Modulul de intrare nu are înregistrare de început de modul, ceea ce înseamnă că nu are formatul corect de modul obiect relocatabil.

— <nume fişier>, PREMATURE EOF
RECORD TYPE xxH, RECORD NUMBER x

Fişierul de intrare se termină înaintea numărului de octeţi specificat de lungimea înregistrării.

— <nume fişier>, PROGRAM EXCCEDS 64K

Modulul în curs de relocare depăşeşte spaţiul de adrese de 64 Ko.

— <nume fişier>, RECORD TOO LONG
RECORD TYPE xxH, RECORD NUMBER x

Lungimea înregistrării fişierului de intrare depăşeşte lungimea maximă permisă pentru înregistrare.

Mesaje de erori nefatale

Mesajele de erori nefatale generate de LOCATE sînt scrise în fişierul hartă (MAP) şi/sau sînt trimise la consolă. LOCATE termină prelucrarea înainte de a reveni la SFDX-18.

— INPAGE SEGMENT COERCED TO PAGE RELTYP

Dacă apare acest mesaj, un segment marcat ca relocatabil în subpagină a trebuit să fie schimbat ca relocatabil în pagină.

— MEMORY CONFLICT FROM xxxxH THROUGH xxxxH

Acest mesaj apare dacă două sau mai multe segmente de program se află în conflict pentru aceleaşi locaţii de memorie. Conflictul este de asemenea semnalat în fişierul hartă.

Acest mesaj este suprimat dacă apare o cerere de MAP.

— UNSATISFIED EXTERNAL REFERENCE AT xxxxH

Acest mesaj este generat la apariţia unei referinţe la un simbol care nu este conţinut la adresa dată în modulul în curs de alocare.

Mesajele de eroare ale comenzii LIB

Toate mesajele de eroare referitoare la ordinele LIB nu sînt fatale deoarece LIB este un program interactiv. Comanda (ADD, CREATE, DELETE, EXIT sau LIST) care a cauzat eroarea este abandonată.

Aceste erori care sînt cauzate de introducerea unei comenzi incorecte sînt urmate de o imagine parțială a comenzii cu (=) în vecinătatea erorii.

— INSUFFICIENT MEMORY

Nu este suficientă memoria disponibilă pentru execuția comenzii.

— INVALID MODULE NAME

<imagine parțială a comenzii>

Un nume de modul este invalid în comandă. El poate avea un prim caracter ilegal.

— INVALID SYNTAX

<imagine parțială a comenzii>

Aceasta este o eroare în comandă. Se datorează următoarelor cauze :

— cuvintele cheie sînt scrise greșit ;
— & este urmat de un caracter diferit de blank ;
— ADD : TO <nume fișier> nu este urmat de <CR> ;
— DELETE : <nume de bibliotecă> (<nume de modul>) nu este urmat de <CR>
— DELETE : <nume de modul> nu este specificat ;
— CREATE : <nume fișier> nu este urmat de <CR>
— LIST : TO <nume fișier> nu este urmat de PUBLICS sau <CR>

— FILE ALREADY EXISTS

Fișierul specificat în comandă CREATE există deja.

<nume fișier>, BAD RECORD SEQUENCE

Fișierul specificat în comandă are o secvență de înregistrare incorectă. Poate să nu fie terminat cu o înregistrare EOF sau se apelează o funcție ADD a unui fișier care nu este obiect sau nu este de tip bibliotecă într-o bibliotecă.

— <nume fișier>, CHECKSUM ERROR

Fișierul specificat conține o înregistrare care are o sumă de control invalidă.

— <nume fișier>, DUPLICATE SYMBOL IN INPUT

Se apelează o funcție ADD cu un fișier care conține un simbol PUBLIC care există deja în bibliotecă.

— <nume fișier>, ILEGAL RECORD FORMAT

Fișierul specificat în comandă are un format ilegal. Fișierul obiect poate conține un nume care are mai mult de 32 caractere. Fișierul poate conține înregistrări într-o ordine incorectă.

— <nume fișier>, <nume modul> NOT FOUND

Se încearcă ștergerea unui modul care nu există. Probabil numele modulului este greșit.

— <nume fișier>, NOT LIBRARY

Fișierul specificat nu este de tip bibliotecă.

— <nume fișier>, OBJECT RECORD TOO SHORT

Fișierul specificat conține o înregistrare de lungime insuficientă.

— <nume fișier>, PREMATURE EOF

Înregistrarea EOF se găsește mai înainte decît este specificată de lungimea fișierului indicat.

— LEFT PARANTHESIS EXPECTED

<imagine parțială a comenzii>

Lipsește o paranteză "(" în comandă.

— <nume modul> ATTEMPT TO ADD DUPLICATE MODULE

Modulul specificat există deja în bibliotecă.

— MODULE NAME TOO LONG

<imagine parțială a comenzii>

Numele modulului specificat depășește 31 caractere

— RIGHT PARAMTHESIS EXPECTED

<imagine parțială a comenzii>

Lipsește o paranteză “)“ în comandă.

— <simbol> PUBLIC SYMBOL ALREADY IN LIBRARY

Se încarcă apelarea funcției ADD cu un modul care conține un simbol PUBLIC, care există deja în bibliotecă.

— “TO“ EXPECTED

<imagine parțială a comenzii>

Fișierul bibliotecă nu este specificat în comanda ADD

— UNRECOGNIZED COMMAND

S-a întîlnit o comandă ilegală sau prost scrisă (ADD, CREATE, DELETE, EXIT sau LIST).

Anexa 3

Sumarul comenzilor CREDIT

În mod ecran:

CTRL/A — Inserează text pînă la următorul CTRL/A; şterge restul ecranului pentru vizualizarea textului introdus.
CTRL/C — Inserează un caracter în poziţia cursorului; deplasează restul liniei la dreapta cu o poziţie.
CTRL/Z — Şterge text pînă la următorul CTRL/Z.
CTRL/D — Şterge caracterul din poziţia cursorului; deplasează restul liniei la stînga cu o poziţie.
CTRL/V — Afişează pagina curentă; trecere din mod comandă în mod ecran.
CTRL/N — Afişează pagina următoare.
CTRL/P — Afişează pagina precedentă.
CTRL/F — Expandează şi execută macrocomenzi.

Pentru deplasarea cursorului se utilizează tastele asignate în acest scop (Se poate executa în mod comandă ? A pentru a afla asignarea curentă).
Pentru a modifica text se introduc noile caractere peste cele vechi.
Trecerea în mod comandă se face cu CTRL/SHIFT/M sau Home.

În mod comandă:

A *cod=valoare nouă...*	Adaptează CREDIT la terminal.
? A	Afişează valorile curente ale funcţiilor modificabile.
B	Indicatorul la începutul fişierului extern.
CR, CW	Închide fişierul extern
DC[n | — n |*marcaj*]	Şterge n(—n) caractere sau pînă la *marcaj*.
DL[n | — n]	Şterge n(—n) linii.
EL	Ieşire din bucla de comenzi.
EQ	Ieşire din editor fără salvarea modificărilor.
EX[*nume fişier*]	Ieşire din editor cu salvarea modificărilor.
F/*şir*/[n | — n | *marcaj*]	Caută un şir şi deplasează *Indicatorul* acolo.
G[*nume fişier*]	Citeşte *fişier* de comenzi.
H	Afişează sumarul comenzilor.
I/*text*/	Inserează *text*; deplasează ***indicatorul***.
J[n | — n | *marcaj*]	Deplasează *Indicatorul* cu n(—n) caractere sau la *marcaj*.
L[n | — n]	Deplasează *Indicatorul* cu n(—n) linii.
MD *nume* | x	Şterge macrocomanda *nume* sau pe toate (x).
MF *nume* [*arg*, [...]]	Expandează şi execută macrocomenzi.
MS *nume/şir comenzi/*	Defineşte macrocomenzi.
? M	Afişează macrodefiniţiile existente.
OR *nume fişier*	Deschide *fişier* pentru citire.
OW *nume fişier*	Deschide *fişier* pentru scriere.

P[n | — n | *marcaj*]   Tipărește n(—n) linii sau pînă la *marcaj*.

QF ; { *com* | <*com*> {   Execută { *com* | <*com*>{ dacă semaforul query=fals.

QT ;} *com* | <*com*> }   Execută } *com* | <*com*> } dacă query=adevărat.

QU   Poziționează semaforul Query.

R[n]   Citește și inserează la *Indicator* n linii din fișierul extern deschis pentru citire.

S/*text1*/*text2*/[n | — n | *marcaj*] — Înlocuiește *text1 cu text2* în zona *Indicator : marcaj* sau *Indicator* : | ±n.

SQ idem ca S   Înlocuiește numai dacă răspunsul este Y, y.

TD n   Șterge marcaj (n=0 : 9).

TS n   Definește marcaj (n=0 ;9).

U/*text*/   Afișează text la consolă.

W[n]   Scrie n linii în fișierul extern.

XC *marcaj*, { *n* | — *n* | *marcaj* }Copiază text de la *marcaj* la *Indicator*.

XM *marcaj*, { *n* | — *n* | *marcaj* }Mută (cu ștergere) text de la *marcaj* la *Indicator* (n, —n linii sau pînă la *marcaj*).

YF ;{ *comandă* | <*comenzi*> }— Execută *comandă* | <*comenzi*> dacă semaforul **Yes** este fals.

YT ; { *comandă* | <*comenzi*> }— Execută *comandă* | <*comenzi*> dacă semaforul **Yes** este adevărat.

Bibliografie la volumele 1 şi 2

1. A. Petrescu şi colectiv IPB, ICE — FELIX M-18, *Manual de utilizare*, Ed. I 1979, Ed. a II-a 1982, Tipografia Institutului Politehnic Bucureşti.
2. A. Petrescu, C. Constantinescu, M. Mihalcea, FELIX M-18, *Biblioteca de subprograme ştiinţifice*, Manual de utilizare, 1982, Tipografia Institutului Politehnic Bucureşti.
3. * * * 8080 *Microcomputer Systems User's Manual*. Intel. Corp. 1975.
4. A. Osborne. *An Introduction to microcomputers*. Vol. II, Some real products SYBEX — 1976, Paris.
5. * * * *Intel. Component Data*, Catalog 1980.
6. * * * *AP-28 A Intel. Multibus Interfacing*. In The 8086, Family User:s Manual October 1979. Intel.
7. * * * *Manuale de funcţionare M18*. Întreprinderea de Calculatoare Electronice.
8. * * * *Manuale de utilizare M18/M118*. Întreprinderea de Calculatoare Electronice.
9. * * * *ISIS-II — User's Guide*. Intel. Corp. 1979.
10. ICI — ICRCH — *Studiu privind utilizarea calculatoarelor pentru conducerea proceselor tehnologice în industria celulozei şi hîrtiei*, 1976.
11. Şt. Dumitrescu, I. Dumitrescu, V. Marinoiu, I. Macri, I. Locovei. *Aplicaţii inginereşti ale calculatoarelor* — vol. 2, Ed. Didactică şi Pedagogică, Bucureşti, 1977.
12. E. Niculescu Mizil. *Consideraţii privind utilizarea microcalculatoarelor şi minicalculatoarelor în conducerea proceselor chimice*, BRI vol. II — 4 — 1982.
13. Metodologie ICI 1982.
14. P. Obrocea, D. Popescu. *Consideraţii asupra folosirii factorului la dezincrustarea sulfat a lemnului*, Revista Celuloză şi Hîrtie nr. 2 1978.
15. P. Obrocea, D. Popescu, N. Latcu. *Model autoadaptabil de conducere a procesului de fabricare a celulozei sulfat*, Revista Celuloză şi hîrtie nr. 3 1978.
16. N. Latcu. *Raport de lucru* — CTCE Suceava, 1982.
17. M. Florescu. *Metode ştiinţifice în dezvoltarea industriei chimice moderne*. Ed. Academiei 1974.
18. D. Popescu, R. Măgureanu, N. Latcu, L. Kreindler, D. Lupusoru, C. Marin. *Conducerea cu ajutorul microcalculatorului M-18 a procesului de fabricare a celulozei* — IP. Bucureşti 1981. Sesiune de comunicări ştiinţifice.
19. D. Popescu, N. Turtureanu. *Conducerea optimă a procesului de fabricare a celulozei sulfat*, Revista Celuloză şi Hîrtie nr 1, 1979.
20. M. Găinariu, D. Popescu. *Realizări ale informaticii industriale în judeţul Suceava*. Simpozion ICI „Informatica şi dezvoltarea economico-socială a României", Bucureşti, ian. 1983.
21. Brendan, Davies. Fundamentals of Automatic Test Equipment for pcbs. În : *„Electronic Production"*, vol. 12, nr. 1, ian. 1983, p. 28—34.
22. * * * Non-destructive testing needs. În : *„International Systems"* April 1979, p. 5.
23. Jon Turino, H. Frank, Binnendyk. *Design to test*, Logical Solutions Incorporated, 1982, 96 Sheveen Place, Campbell, California, 95008, U.S.A.
24. * * * Intel-Component Data Catalog 1981.

25. S. P. Morse, B. W. Ravanel, S. Mazor and W. P. Pohlman. Intel Corporation, *Intel Microprocessors 8008 to 8086*, Computer, Oct. 1980.
26. * * * Microprocessor development systems'survey, *System International*, November, 1980.
27. B. E. Gladstone. *Comparing microcomputer development system capabilities*, Computer Design, February 1979.
28. C. Lupu, V. Țepelea, E. Purice. *Microprocesoare* — Aplicații, Editura Militară, 1982.
29. M. Biewer. *Microprocessor Architecture*, Pro-Log Microprocessor User's Guide, 1980.
30. * * * INTEL 8080 Assembley Language Programming Manual.
31. J. F. Vittera. *Handling multilevel subroutines and interrupts in microcomputers*, Computer Design, 1978, 17, 1.
32. L. F. Danaghey, G. M. Bobba, X. K. Rubin. *Decisian-making with flags in process control*, Computer Design, 1976.
33. J. Peatman. *Microcomputer-based design*, Mc Graw-Hill, London, 1977.
34. J. Beaston. Using the 8259 Programmable Interrupt Controller, AP-31, Intel Corp.
35. L. Smith. Using the 8251 USART, AP-16, Intel Corp.
36. A. Ebright. 8255 *Programmable Peripheral Interface Applications*, Intel Corp., AP-15.
37. * * * RMX/80 *Real Time Multitasking Executive*, AP-35, Intel Corp.
38. D. Larsen, P. Rony, M. Dejong, C. Titus, J. Titus. *Microcomputer Interfacing: A demonstration Program for the 8253 Timer*, Computer Design, 1978.

Anexa 1 continuă din vol. I, pag. 288.

Nr. crt.	Denumire (acronim) și scurtă descriere funcțională		Echipament	Elaborator: direcție, compartiment
1	2		3	4
66	DISPECER	Dispecerizarea și conducerea operativă a proceselor de producție continue	M-18 M-118	ICI-DC Laborator 1
67	MICRO-PACK	Bibliotecă de calcule tehnico-științifice	M-18 M-118	ICI-DC Laborator 4
68	MICRO-S	Pachet de programe standard pentru supravegherea și conducerea utilajelor și proceselor tehnologice	M-18 M-11 SPOT-80 SELROM ECAROM	ICI-DC Laborator 4
69	ETXMX	Programe de introducerea de texte de lungime variabilă	M-18 M-118	ICI-DP Colectiv MICRO
70	TRIMX	Program de sortare fișiere tip ETXMX	M-18 M-118	ICI-DP Colectiv MICRO
71	INBIS	Program de introducere interactivă și corectare (editare) a informației de tip lent (editor de texte)	M-18 M-118	ICI-DP Colectiv MICRO
72	INPFAO	Sistem de programe pentru structurarea informațiilor introduse prin INBIS sau alt sistem de introducere de date ; acceptat de specificațiile FAO pentru sistemul de informare documentară AGRIS.	M-18 M-118	ICI-DP Colectiv
73	TRANSC	Program de conversie a fișierelor SFDX-CP/M	M-18 M-118	ICI-DC Laborator 4
74	COMATI	Sistem de programe pentru conducerea automată cu calculator a operațiilor de triere în triajele de cale ferată	M-18 CAMAC	ICI-DC Laborator 4 în colaborare cu MTTc-ICPTT

NOI APARIŢII

În trimestrul IV 1984 urmează să apară următoarele lucrări în seriile din domeniile automatică-informatică-management :

M. Mănescu şi colectiv, Modele de optimizare în industria petrolului, ~600 pag., ~25 lei, Seria *Fundamente.*

I. Văduva şi colectiv, Analiza economico-financiară asistată de calculator, ~500 pag., ~40 lei, Seria *Practică.*

V. Baltag şi colectiv, Sisteme grafice interactive, ~500 pag., ~40 lei, Seria *Practică.*

L. Dumitraşcu, Învăţăm COBOL ... conversînd cu calculatorul, (COBOL pe Felix C 256, pe minicalculatoarele CORAL, Independent şi pe microcalculatoarele Felix M 18, M 18B, M 118), vol. 1 şi vol. 2, ~600 pag., ~50 lei, Seria *Iniţiere,* Ciclul Limbaje universale de programare.

A. Petrescu şi colectiv, Limbajul BASIC şi calculatorul personal AMIC, vol. 1 şi vol. 2, ~500 pag., ~50 lei, Seria *Iniţiere,* Ciclul Limbaje universale de programare.

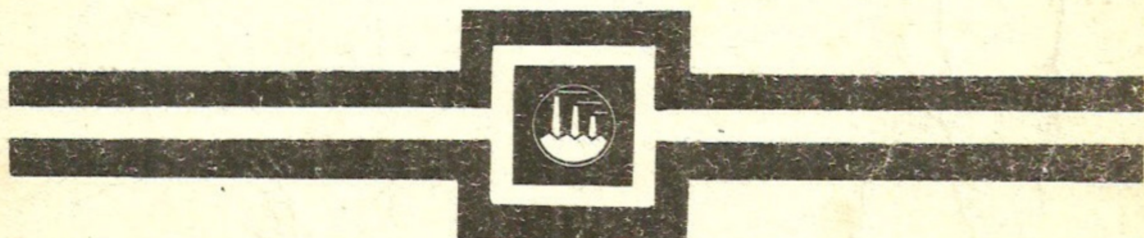
O. Cernian şi colectiv, Limbajul BASIC VS pe calculatoarele WANG, 250 pag., 22 lei, Seria *Practică.*

Colective de specialişti, Roboţii în Japonia. Service pentru calculatoare (AMC 43), noile calculatoare WANG, 240 pag., 22 lei, Seria *AMC.*

Idem, Minicalculator INDEPENDENT (AMC 44), 300 pag., 25 lei, Seria *AMC.*

Idem, Testarea automată. Manual de analiză (AMC 45, AMC 46) ~600 pag., 2 volume, ~45 lei, Seria *AMC.*

Idem, Automatica, informatica şi tehnologiile moderne(AMC 47, AMC 48, AMC 49). Congresul IFAC, Budapesta 1984, 900 pag., 3 volume, ~66 lei, Seria *AMC.*



EDITURA TEHNICĂ

Vol. I și vol. II – Lei 46